Universidade de Aveiro
2018
Departamento de Eletrónica,
Telecomunicações e Informática

MARCOS
PIRES

DESENVOLVIMENTO DE FERRAMENTA INTELIGENTE PARA DETEÇÃO DE PHISHING EMAILS

DEVELOPMENT OF INTELLIGENT TOOL FOR PHISHING EMAIL DETECTION

Universidade de Aveiro
2018

Departamento de Eletrónica,
Telecomunicações e Informática

MARCOS PIRES

DESENVOLVIMENTO DE FERRAMENTA INTELIGENTE PARA DETEÇÃO DE PHISHING EMAILS

DEVELOPMENT OF INTELLIGENT TOOL FOR PHISHING EMAIL DETECTION

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica da Doutora Petia Georgieva, Professora auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus pais.

o júri / the jury

presidente / president

Prof. Doutor Luís Filipe de Seabra Lopes
professor associado da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Ivo André Soares Pereira
professor adjunto convidado do Instituto Politécnico do Porto

Prof. Doutora Petia Georgieva Georgieva
professora auxiliar da Universidade de Aveiro

agradecimentos /
acknowledgements

Primeiramente agradeço o apoio dado pelos meus pais e irmãos que foi incessante desde que sou gente. Fico grato pelo companheirismo oferecido pelos meus colegas e amigos do grandioso curso de MIECT, sem eles a vida teria sido muito mais difícil nestes cinco anos que passei na Universidade de Aveiro. Refiro também todos aqueles que me receberam de braços abertos no CUFC, a minha casa fora de casa. À minha orientadora, a professora Petia Georgieva, fico grato pela completa disponibilidade e pela motivação que demonstrou e transmitiu. Obrigado também ao pessoal da E-goi que fez o meu estágio valer a pena pelas pessoas que conheci e os momentos que vivi, bem como às ajudas que me poderam oferecer nesta dissertação e financeiramente. Neste último anos também comecei uma caminhada no grupo de voluntariado de Teresa de Saldanha de Aveiro, que me tem sido mais importante do que alguma vez poderia ter inicialmente especulado.
Estes cinco anos foram determinantes tanto para a minha vida pessoal como profissional, e foram muitas as pessoas que participaram neste meu crescimento. Algumas poucas cresceram na minha consideração ao ponto de hoje as considerar indispensáveis. Os meus agradecimentos a estes meus ídolos serão recebidos de uma forma muito mais pessoal, simplesmente por realmente o merecem.
Mas principalmente agradeço a Deus pela força e paciência que me deu.

Palavras Chave

emails de phishing, aprendizagem automática, seleção de características, random forest.

Resumo

Emails de Phishing são um tipo de ataque comum na internet que resultam no roubo de informação confidencial de utilizadores como contas bancárias, dados privados, logins pessoais ou de identidade. O objetivo desta tese de mestrado passou por desenvolver uma ferramenta inteligente baseada em abordagens com aprendizagem automática para filtrar este tipo de emails malignos. O projeto foi feito em cooperação com a E-goi, empresa de automação de marketing multicanal. A primeira etapa do projeto foi a de selecionar aspectos característicos dos emails de modo a poder diferenciar entre emails de phishing e normais. O conjunto final destas características foi escolhido depois de um estudo minucioso da literatura e das necessidades da empresa. O passo seguinte foi a escolha de um algoritmo eficiente para a deteção de emails de phishing. Como a tarefa foi considerada um problema de classificação, vários algoritmos de aprendizagem automática foram testados (SVM, DT, Random Forest, Boosted Trees). Um grande desafio que foi deparado durante o desenvolvimento foi o da falta de dados categorizados, mais especificamente do tipo de phishing. Para tentar contornar o problema, o sistema de detação de phishing foi construído com ajuda de dados (emails) publicamente disponíveis. De modo a facilitar a implementação de um protótipo na empresa E-goi, foi desenvolvida uma ferramenta web para categorizar a colecionar emails. Este sistema permite a pessoal autorizado da empresa a fazer a categorização on-line de emails adquiridos.

Keywords phishing emails, machine learning, feature selection, random forest.

Abstract

Phishing emails are a very common attack on the web, that results in the theft of confidential user information such as bank accounts, private data, personal logins or of identity. The goal of this master thesis was to develop intelligent tools to filter out the emails with such malign intent, based on machine learning approaches. The work was done in close collaboration with a multichannel marketing automation company of name E-goi. The first stage of the project was to select appropriate features able to discriminate between ordinary and phishing emails. The final feature set was chosen after a comprehensive study of the literature and the particular needs of the involved company. The next step was to choose an efficient algorithm for phishing emails detection. Since this task was considered as a classification problem, a number of machine learning classifiers were tested (SVM, DT, Random Forest). A major challenge during development was the lack of sufficient labeled data, particularly regarding the class of phishing emails. To get around this, the phishing detection system was built based on a collection of samples (emails) from different publicly available data sets. In order to facilitate the implementation of the phishing detection prototype in the company E-goi, a web tool was developed to create a home-made labeled data set of emails. This system allows authorized company personnel to label on-line each obtained email.

Contents

List of Figures

List of Tables

Glossary

URL	Uniform Resource Locator
TLD	Top-level domain
IP	Internet Protocol
DNS	Domain Name System
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
TLS	Transport Layer Security
HREF	Hypertext Reference
SMTP	Simple Mail Transfer Protocol
TF-IDF	Term Frequency - Inverse Document Frequency
SOM	Self Organizing Map
SVM	Support Vector Machine
BSVM	Biased Support Vector Machine
K-NN	K-Nearest Neighbor
LR	Linear Regression
CV	Cross-validation
SMS	Short Message Service
MIME	Multipurpose Internet Mail Extensions
CSV	Comma Separated Values
RBF	Radial Basis Function
OOB	Out-Of-Bag
EML	Electronic Mail
PCA	Principal Component Analysis
RAM	Random-Access Memory
CPU	Central Processing Unit
ROC	Receiver Operating Characteristic
AUC	Area Under the Curve
RF	Random Forest
MSE	Mean Square Error
DT	Decision Tree
API	Application Programming Interface

CHAPTER 1

Introduction

Phishing emails are a common attack on the web that results in the theft of confidential user information such as bank accounts, private data, personal logins or identity. These attacks have a heavy impact on the web and on the global economy, as billions of dollars are reported stolen every year.

Finding an automatic solution for this problem is a difficult task because phishers are very creative, and often it is hard even for a human to differentiate between legitimate and malign content. Many of the existent techniques rely on available and regularly updated black Uniform Resource Locator (URL) lists to filter out harmful messages. Others use knowledge obtained from past attacks to create empirical rules that decide if a new mail is phishing or not. Over the last years machine learning algorithms have been gaining popularity as a mechanism to find hidden relationships between the emails.

The objective of this thesis was to develop an intelligent tool for phishing detection, based on the principles of machine learning, suitable for implementation in the multi-channel marketing automation company E-goi. The system was intended to work as a filtering mechanism to help deal with the high amounts of emails that the company receives daily. The goal was to reduce the amount time spent by workers that had to go through hundreds of emails everyday to verify their authenticity, when only a small portion would actually need to be flagged as phishing.

A major issue was the lack of labeled dataset. This was because no previous attacks were saved, and new phishing attempts were scarce and mixed with the legitimate emails. Hence, an additional objective (not envisaged at the beginning of the project) was defined, namely to develop a suitable web service to label the incoming emails, and therefore create a sufficiently large dataset to allow a reliable study to be done.

The content of this document is organized in the following chapters. In Chapter 1 the motivation and the objectives of this work are outlined. Overview of phishing

detection principles and related works is done in Chapter 2. The proposed solution is described in Chapter 3. It has three main stages. The first one is the creation of a reliable dataset with similar characteristics as the email structure and content of the company‘s clients. The second stage is the extraction of the most reliable features to discriminate between regular and phishing emails. Two approaches are followed: i) Binary features (0 or 1) - often suggested in the revised phishing detection literature; ii) Bag of words - more typical features in the text mining context. The third stage is the search for an efficient algorithm for phishing detection among well known supervised learning models. In Chapter 4 the back and the frond end of the email labeling tool is detailed. Finally, in Chapter 5 a summary of the main contributions of this work and suggestions for future developments are presented.

CHAPTER 2

Background and Context

This chapter is dedicated to exposing a review of the literature and give a context to this dissertation. Section 2.1 describes the problem. Section 2.2 describes the gathered information about the different methods that are currently being used. Section 2.2.1 gives an insight of what Blacklists are and gives examples of available services. Section 2.2.2 describes whitelists and proposed methods for their implementation. In section 2.2.3 some approaches with different heuristics are presented. Section 2.2.4 describes the basics of machine learning, including some of the most used algorithms in the context of phishing detection and offers some examples. Section 2.2.5 explains the method of comparing webpages visually. Section 2.2.6 briefly explains the hybrid approaches. And lastly, section 2.2.7 explains the specific case of smishing.

2.1 The Phishing Problem

Phishing is a type of cybercrime where the attackers pose as a legitimate entity, with the intent of stealing sensitive personal information from unsuspecting users, through the use social engineered techniques. These techniques, according to Criddle[1], can be defined as types of manipulations used on users to make them believe they are safe, when the opposite is true. Phishing attacks can be done by email or any other social network or communication channel, hence the different names. They can cause permanent damage to legitimate businesses, the financial loss is in the order of the billions of dollars, and the situation is only getting worst.

2.1.1 Types of Phishing attacks

There are several types of phishing attacks, according to Shi et al. [2] they can take many forms, the most common being the following.

Clone Phishing

In these attacks, emails from a legitimate entity is obtained, cloned and changes are made over it. Some of these changes may be that links are changed to redirect to malicious websites, or email addresses are spoofed so the sender appears to be original, or even the attachment of malicious content.

The target is not specifically defined, they are usually sent randomly to several different destinations, in the hopes of fooling anyone who receives them.

Spear Phishing

These attacks have a specific target group, where only a pre-determined group of people, with a common characteristic (e.g. people from the same organization), will receive deceitful content.

Phone Phishing

Phone phishing consists of messages appearing to be from a reliable source requesting users to contact a determined phone number for some seemingly reasonable motive. The information theft occurs during the call.

2.1.2 Damage Report

According to Internet Crime Report from 2016[3], every year the average number of complaints regarding phishing attacks is of about 280.000, with an estimated loss of 1.33 Billion dollars in 2016. Note that these values are only of reported occurrences, real values are estimated to be much higher.

The motivation behind this type of attacks can be for financial gain, the stolen information can be bank credentials; can be identity hiding, where the phishers might sell the information to others who want to hide their identity; or even for fame and notoriety.

2.2 Literature Review

To try and solve the phishing problem, many methods have been proposed and implemented. Such methods can be summarized into five different categories: blacklisting, whitelisting, machine learning, Heuristic and visual similarity. For each, a brief description of the category and some real examples are given.

2.2.1 Blacklisting

Blacklists are lists of URLs of known malicious sites. By querying such lists with URLs which intent is unknown, it is possible to assert if it is in fact a phishing attempt or not. This method is fairly simple to understand and implement as the algorithms only

need to compare strings or sub-strings. But there is a great problem that invalidates this solution in many cases, as it is extremely vulnerable to zero-day attacks. These are attacks where the malicious content, in this case the URL, is new and was not yet reported and added to the blacklists. Therefore, any query thrown at those lists will receive a false negative response.

Some good examples of a popular blacklist services available are Phishtank, where users submit suspected phishing URLs and a subsequent vote by other users will assert if those URLs have a malicious intent or not, and Google safe browsing[4], a continuously updated list where users can submit potential new malicious webpages or an automated detection of websites occurs, but Google's software will ultimately classify the page as safe or not.

PhishNet [5]

Phishnet is a system that tries to improve the effectiveness of blacklists. It uses blacklists and some heuristics to try and predict new phishing URLs. This is a possible solution to the problem of the exact matching method that is traditionally used for blacklists that makes it easy for attackers to exploit, as the slightest alteration on a word might go unnoticed by the user and by the algorithm. Two components exist, the first being the prediction of malicious URLs, and the second the approximate matching of URLs.

The first component, **predicting malicious URLs**, examines existing blacklists and combines pieces of the URLs in those lists automatically, with some heuristics, to generate new URLs. After this the generated URLs need to be validated.

Five heuristics used:

- Replacing Top-level domains (TLDs) - Try to find variants of the original blacklist elements by changing only the TLD of the URLs.
- Internet Protocol (IP) address equivalence - The premise is that phishing campaigns hosted by the same IP address might share the directory or path structure as well. URLs with same address are clustered together and new URLs are created by applying all possible combinations between hostname and paths.
- Directory structure similarity - The premise is that URLs with a similar directory structure may also have similar set of file names. These URLs are clustered and new URLs are created from the different combinations of filenames among URLs within a cluster. For example "if www.abc.com/online/signin/ paypal.htm and www.xyz.com/online/signin/ebay.htm are two known phishing URLs then our heuristic predicts the existence of www.abc.com/online/signin/ebay.htm and www.xyz.com/online/signin/paypal.htm"[5].
- Query string substitution - The use of query strings are a way to change the URL without the destination being changed, this is a weakness of the exact matching

method used in blacklists that can be exploited by attackers. This heuristic tries to stop those exploits by creating combinations of query strings between URLs already in the blacklist. For example if there are two URLs, `www.abc.com/online/signin/ebay?XYZ`, and `www.xyz.com/online/signin/paypal?ABC`, two new URLs are created, `www.abc.com/online/signin/ebay?ABC` and `www.xyz.com/online/signin/paypal?XYZ` [5].

- Brand name equivalence - It is a know fact that attackers target legitimate brand names using same URL structure method. This heuristic creates candidate URLs by replacing brand names present by all the other known brand names.

After creating the candidates for new phishing URLs a validation must be done to filter out the ones that are non-existent or legitimate. This validation process consists in a first Domain Name System (DNS) lookup to determine if the URL exists. If so a connection to the server is done and a Hypertext Transfer Protocol (HTTP) GET request is sent to get content from it. If the HTTP header from the response is a status code of a successful response, a similarity detection tool is used to compare the parent URL to the child. If they are found to be similar, the child is considered a phishing URL and is added to the blacklist.

The second component, **Approximate matching**, does an approximate match of new URLs to the ones in the blacklist, using methods based on regular expression similarity and hashmaps to try and catch syntactic and semantic variations, so that even if a new URL is not exactly matched to any entry of the blacklist, it can still be caught and labeled as phishing.

The approximation relies on four different aspects of an URL that correspond to module, from each a score is calculated. If the value computed from all the scores is greater than a pre-configured threshold the URL is considered a possible threat.

The four modules used for the approximate matching are the following:

- Matching IP address - score is calculated from the number of IP addresses from the blacklist that match the IP address of a new URL.
- Matching hostname - score is calculated from the number of blacklist entries that match the hostname of a new URL.
- Matching directory structure - This module creates a hashmap with the directory structure as keys, and the number of phishing URLs that contain the same structure as correspondent values. The score is calculated by the value stored in the hashmap where the key is the structure of the new URL.
- Matching brand names - From the pathnames and query strings of the URLs in the blacklist brand names are extracted and a frequency score is attributed to each brand. Brand names are also extracted from new URLs the same way, and the score for a given URL is the frequency score of the extracted brand name.

Observation of the results for the heuristics of the prediction of malicious URLs shows that every new URL that is tested would have a very polarized classification, meaning that they are either very similar or very dissimilar to the URLs in the blacklist. The ones that have a similarity higher than a threshold would be considered as new phishing URLs. For the approximate matching, a trade-off was evident between false positives and false negatives, a higher threshold would yield more false negatives and a lower would yield more false positives. Overall Phishnet has low false positives and is very effective in adding malicious URLs to the blacklist.

A Phishing Sites Blacklist Generator [6]

This technique uses the advantages of blacklists and tries to go further by creating an automatically and locally updated blacklist of fishing sites. What this means is that the authors explored the fact that every webpage is associated to a website, and that in most websites this relationship is evident through the use of logos. If phishing websites steal the logo from the legitimate business to gain credibility, searching these for the URLs of these images with a search engine, such as Google, to obtain the rightful domain, might be a good way to reveal the website's true intent. In this search only the top 10 results are considered. If an URL goes through the algorithm and is considered phishing, it is added to a blacklist that is used to filter out phishing attacks.

The algorithm proves to be very efficient, with a high accuracy for phishing detection and a considerably low amount of false positives. However it is a very time consuming approach if implemented locally and not on an email server.

2.2.2 Whitelisting

While in blacklists only malicious URLs are considered, in whitelists only legitimate ones are saved and stored. Any webpage that is not listed by these lists is considered suspicious. This solves the issue of zero-day attacks but leads to other problems. The biggest being the fact that it is impossible to have listed all existing legitimate websites. This method may cause a high amount of false positives as new legitimate sites might not have been yet included.

The use of automated individual whitelists was proposed by Cao et al. [7]. They automatically maintain a list of the commonly visited login interfaces of a single user. The number of visited websites of any user is assumed to be small, according to Florencio and Herley [8], meaning that the locally created whitelists are very small but very effective. When the user visits a webpage that requires a login and it is not on the list, a warning of a possible phishing attack is shown.

2.2.3 Heuristic Methods For Phishing Detection

The heuristic methods employ the knowledge obtained from real phishing attacks analysis to help create a mechanism to detect or prevent such attacks. The heuristics consist in obtainable recurrent features that characterize phishing attacks. This type of methods offer a better solution than the previously described, as they proved more efficient than blacklists when dealing with zero-day attacks, and it does not yield as many false positives as whitelists. One downside is the lack of adaptability, as it is a static method, and the fact that it can have a considerable amount of false negatives[9][10].

SpoofGuard [11]

Spoof is the name of any attempt that seeks to impersonate someone or something they are not. SpoofGuard is a plug-in for web browsers to monitor a user's activity while browsing. The idea was to calculate a value, of name spoof index, that indicates the likelihood of a visited website being a phishing attack. The user is warned about the possible attack if the spoof index exceeds a threshold defined by the user. The index is calculated using information about the website, and examination of outgoing post data. The more suspicious characteristics there are, the higher the spoof index.

Upon visiting a webpage, SpoofGuard extracts and analyses these aspects:

- Domain check - It asserts if the user has previously visited the domain.
- Image-domain associations - Checks if the visited webpage was referenced from an email site.
- URL check - Examins if the URL has misleading elements, and if a page domain is relatively similar to a commonly spoofed or regularly visited domain.
- Image check - When an image is present it is compared to a database of images and it's legitimate domain is extracted, then this is compared to the domain where the image was first obtained, this asserts if the image was stolen or not.
- Link check - Present links are put through a process similar to the URL check.
- Password check - When a login is required or if sensitive information is asked, the plug-in checks if https is in use and if the certificate check succeeds
- Outgoing password check - There is also a mechanism that warns users if the login information they inserted on a regularly visited webpage is in a domain different than usual.

This heuristic had a considerable success and with no significant performance penalties for the machines where the tests took place. This is not a solution for phishing detection on emails as it is dedicated to function as a plug-in on a web browser, but some procedures used can inspire new methods for prevention.

Phishwish [12]

Phishwish is a phishing email filter which goals are to identify zero-day attacks, to create a rule based system to simplify the detection process, and to make such process as efficient as possible with the least amount false positives as possible. The algorithm is capable of processing emails with either plain text or Hypertext Markup Language (HTML).

To categorize emails as either malicious or regular 11 rules are used. Each has an associated weight and flag. Every rule will produce a value and the sum of said values will determine the outcome of the algorithm. The higher the value the more likely it is malicious. The rules are the following, if the rule applies it is an indication of phishing:

- Rule 1 - Sender email does not redirect to login page of the legitimate business, this is verified through a search engine.
- Rule 2 - When an email has HTML content, and there are URLs displayed that use Transport Layer Security (TLS) (a cryptographic protocol) but their respective Hypertext Reference (HREF) does not.
- Rule 3 - The login URL is not a domain name, but is in the form of an IP address.
- Rule 4 - The business name is present in the login URL but is absent from the domain.
- Rule 5 - The domain displayed by a URL is not the same as the domain specified by the HREF.
- Rule 6 - The legitimate business name is not included in the Simple Mail Transfer Protocol (SMTP) header.
- Rule 7 - Assert if the domain of non-image URLs present in the email are different from the domain of the login URL.
- Rule 8 - Assert if the domain name registrant, i.e. the entity registering the domain name, of the non-image URLs present in the email are different from the domain name registrant of the login URL.
- Rule 9 - Assert if the domain of URLs of images present in the email are different from the domain of the login URL.
- Rule 10 - Assert if the domain name registrant of URLs of images present in the email are different from the domain name registrant of the login URL.
- Rule 11 - Webpage is not accessible.

There were considerable problems of false positives and the detection of zero-day attacks was not as successful as the authors intended. Overall it is a solution with many faults and low flexibility but is a good example of a rule based approach.

CANTINA [13]

CANTINA is an approach that adapts an information retrieval algorithm known as TF-IDF to phishing detection. A more detailed description of this algorithm is present

in section 2.2.8. The output of this algorithm are the most characteristic words of the corpus.

The idea behind CANTINA was based on the assumption that phishing sites usually try to create a webpage very similar to the one owned by the legitimate business. This means that by using Robust Hyperlinks (a method to fix broken URLs by adding "signature" words so that if the address-based portion of an URL fails, the content-based signature can be used to try and locate the original document by querying the signatures to a search engine) with the proper signature words it would theoretically be possible to find the genuine login page, as the words would be frequent in the current page but rare across the web. The signatures are keywords extracted with a TF-IDF algorithm from the phishing page, that usually are the names of the businesses being spoofed.

This approach only takes into consideration the five words with the highest TF-IDF weights and then searches for them using Google. The login page is considered phishing if there was no match between the domains of those pages and the login page in the top N results of the search.

In order to reduce the amount of false positives some additional rules were implemented:

- Age of Domain - Phishing sites tend to have a recent creation date, so the age of the domain is considered as a good indicator, for this WHOIS search was used. This method is powerless to attacks where the legitimate site was compromised, as their domain will be considered as legitimate.
- Known Images - Nine logos of popular websites are saved locally so that if a suspicious page has an image similar to one of them, but is in a different domain, the algorithm classifies it as phishing. This process is very similar to one used in SpoofGuard[11].
- Suspicious URL - The URL is checked for suspicious characters, namely "@" and "-" due to their special meanings in this context. This process is very similar to one used in SpoofGuard[11].
- Suspicious Links - The process of checking URLs is used to verify the links present in the page.
- IP Address - Check if the domain name is in the form of an IP address. This process is very similar to one used in Phishwish[12] and PILFER [14]
- Dots in URL - Binary values, where the number of dots in the URL is counted, as experience showed that phishing URLs tend to have a higher number than legitimate ones. The amount considered to be too much to be acceptable is greater than four. This process is very similar to one used in PILFER[14].
- Forms - HTML code is examined for tags that allow input of text with suspicious labels, such as "credit card" and "password", as this is the information that the

attackers want.

The results of this approach were considerably good, with high true positives and low false positives. The number of false positives was successfully reduced with the heuristics described, but true positives decreased as a consequence. A major defect is that this method is vulnerable to image-instead-of-text attacks, as the extraction of keywords will not be as effective.

2.2.4 Machine Learning

Nowadays phishing attacks are becoming increasingly automated, this shift makes traditional methods much less efficient in detecting new attacks. To solve this, more adaptive mechanisms are being implemented, namely with machine learning algorithms, as they can detect hidden patterns and feature correlations that would go unnoticed by other methods.

There are many advanced machine learning algorithms that can be used to classify attacks as phishing or regular. The unsupervised models organise the data in groups in accordance to the various features. Its purpose is the extraction of a structure from the dataset without no knowledge about the to which the data belongs to[15]. Whereas supervised models try to classify the data within the available classes. In phishing it can be considered the existence of only two separate classes, phishing and non-phishing. It was observed that most of the proposed machine learning methods go for a supervised approach, as it tends to yield better results[15].

In every machine learning model some steps have to be taken in order for them to function correctly. Initially the proper dataset must be chosen, and a data preprocessing sometimes is required to correctly format the data. This preprocessing might include the removal of noise, a normalization process and possibly other actions that can be applied, according to the problem it is proposed to solve. The original dataset should be separated into training, validation and testing sets, Only then the machine learning algorithms can be trained with the training set, then the algorithm that performed the best with the validation set is chosen as the one to be used to solve the problem. Some algorithms need some parameters to be chosen, the validation set can also be used to find the values for the parameters that produce the best generalization of the problem. The training and validation sets are then combined to train the chosen model one final time, and the testing set is used to evaluate the final performance. Bellow some of the observed algorithms used in different approaches are briefly described.

Linear Regression (LR). [16] attempts to find the hidden relationship between two variables by defining a linear equation that best fits their relationship. These two are the *independent* variable, or $\mathbf{X}$, and the *dependent* variable, or $\mathbf{Y}$. The equation produced

by this model is $\mathbf{Y} = \mathbf{a} + \mathbf{bX}$. $\mathbf{a}$ is the value of $\mathbf{Y}$ when $\mathbf{X}$ is zero, and $\mathbf{b}$ is the slope of the produced line.

K-NN. [17] is an clustering algorithm that labels new points by finding and analyzing the labels of the $\mathbf{K}$ nearest points from the training data. This model expects data with similar characteristics to have the same labels.

It is specially a useful when there is no knowledge about the data distribution.[18]

Decision trees. [19] have a structure similar to flowcharts, where an attribute is tested in every node. A branch connected to a node is the outcome of the test from the same node, and the leafs of the graph are the final classifications. The path from the first node to a leaf represents the classification rules.

This classifier is able to process data of numerical,nominal or textual type, it is also robust to noise. It offers a good generalization for the prediction of classes new entries, with a high performance contrasting to the low computational effort needed, although this model is not so efficient when high dimensional data is considered.

The process of building a decision tree usually takes a considerable amount of time. The strategy followed is *divide and conquer*, meaning that the nodes test the features of a new entry sequentially until a leaf is reached, giving a prediction to its class. The method works well when there are attributes with high relevance, but is not that efficient when features have complex interactions. Some major problems are the error propagation that occurs through the tree, and the need for a proper pruning of the tree, as it can easily overfit. To solve the last problem random forest, which is an ensemble of threes, is a more appropriate method.

Random forests. [19] are several trained decision trees, where the prediction of the outcome is the most voted class among all the trees. It is scalable, robust to noise, does not overfit, fast and easy to understand and visualize the results. One problem is that with a high number of trees in the forest, the model can hardly offer a real-time answer.

Boosted Trees. [20] is a supervised classifier based on Gradient boosting, a technique that creates an ensemble of classifiers that function together as just one prediction model. Tree Boosting is composed by an ensemble of regression trees, i.e. decision trees with continuous values. This model can produce great and robust results, and is very efficient for noisy datasets.

Bayesian Networks. [19] is a graphical model that represents the dependencies between features. It is a model that needs fitting. It is not very efficient when the dataset has a

high dimensionality, as the network would require too much memory and time to work properly.

Naive Bayes. [19] is an supervised classifier and is a type of Bayesian network with only one parent node and many child nodes, where it is assumed that there is a strong independence between these last nodes, this is called *class conditional independence*. If this assumption is found to be true to a certain example, the model converges fast to the class prediction. The training process takes relatively low computational time. The value that results from the prediction of a class is a probability. This algorithm works best when there is a clear independence between features.

Support Vector Machine (SVM). [19] is a supervised non-probabilistic binary classifier, where a hyperplane or a set of hyperplanes are created for classification of data. An SVM maps the original feature space into a higher dimensional kernel space in order to find the hyperplane that best fits the classification problem, i.e. that separates the training data in two classes with the maximum margin possible. This margin is defined by a subset of training data points, called support vectors, making it more memory efficient. The larger the maximum margin, the better generalization is achieved. The prediction works by mapping the new entries into the kernel space, and then determining to which side of the margin it belongs to. The choice of the kernel function is crucial for making the method computationally efficient.

Although it is a complex algorithm, it offers good generalization for high dimensional spaces and, with the proper kernel, it is useful for cases where there are more dimensions than number of samples. It offers high accuracy and prevents some common overfitting issues, the performance independent from the size of the dataset but dependent from the number of training cycles. Some major setbacks are the low training process speed and the dependency on the right choice of parameters.

Biased Support Vector Machine (BSVM). [15] is a type support vector machines for hard classification problems where the idea is to decompose the problems into subproblems. It works well with unbalanced data.

Neural Networks. [19] is an algorithm inspired by the structure of the human brain and it's ability to learn. Through the training process a highly interconnected network constituted by processing elements (known as neurons) is obtained, with this and a set of inputs a set of outputs is obtainable. Some configuration are needed, one can stop changing the configurations of the network when the results seem appropriate.

Multi-layer Perceptron. [15], a supervised method, is the most used neural network and is a collection of perceptrons. The perceptrons are the simplest form of a neural network with nothing but one single neuron where the weights of the features are adjustable. This model is useful for complex problems, is robust to noise, but is very hardware and time demanding when training. In order for the model to work best the correct size of the hidden layer must be chosen. That is a hard task, as hidden layer that is too big or too small may lead to inappropriate results. The model is very sensitive to the parameters that are chosen for it.

Self Organizing Map (SOM). [15], an unsupervised algorithm, is a type of neural network that organizes the data in clusters, through a competitive process, and maps it onto a two dimensional map, reducing the dimensionality. New data is mapped as well and is classified by which node is the closest to it. This model is useful for dimensionality reduction for better interpretation of the data.

K-Means. [15] is an unsupervised model that organizes the data into K clusters, hence the name, where every entry is assigned to the cluster with the nearest mean. The mean of a cluster is the mean point between all the points assigned to that same cluster. The result is a partition of data into Voronoi cells with somewhat similar sizes. This algorithm is computationally demanding, but with the right heuristics a local optimal can be obtained much faster.

Detecting Phishing Emails Using Hybrid Features[21]

This method proposes the creation of a robust classifier using hybrid features, where the best ones are selected through an information gain evaluator.

It was found that different types of features support each other in the detection of phishing, and that few approaches consider the structure and orthographic features of a phishing email. Three types of features, that had to be extracted, are described below.

- Content features - domain specific keywords that can be used to identify semantic contexts.
- Orthographic features - text elements that define the style of the presented page. These elements give functionality to words or sentences, some examples are HTML features, scripts or images.
- Derived features - obtained from existing content or orthographic features, one example is the similarity between visible and hidden links.

Taking into account the three types of feature, seven different features related to emails are used:

1. Links - number of links.

2. Nonv_links - number of invisible links.
3. Nonmatching URLs - binary value that asserts if the hidden URL is the same as the visible.
4. Forms - existence of forms.
5. Scripts - type of script present, where 0 is none and from 1 to 6 are different types.
6. Body Blacklist Words - amount of appearances of predefined words that might indicate phishing attack, a small blacklist of words, in the body of an email.
7. Subject Blacklist Words - amount of appearances of predefined words that might indicate phishing attack, a small blacklist of words, in the subject of an email.

Before classification the features are normalized in order to belong to similar range of values.

After obtaining the normalized features, a preliminary classifier is created, the results are used to assert the importance of the features, using an information gain algorithm. With this information only the more relevant features were used to effectively train the classifiers, removing possible noise. The classifiers used were five and they were decision tree, random forest, multi-layer perceptron, naive bayes and support vector machine. After the training and testing processes of all the machine learning algorithms the one that showed the best results was the decision tree.

The final product was a decision tree classifier capable of removing redundant features, trained recursively by different datasets, that ultimately has a very high phishing detection accuracy.

Classification of Phishing Email Using Random Forest Machine Learning Technique[22]

In this method[22] some characteristics of known methods influenced the objectives of this tool. It was pointed out that models that use visual processing tend to be time consuming and require additional space in order to function. Another method that is used is querying over the network to obtain features, this process might lead to significant run time increases. To avoid any of this problems it was proposed a method in which all features are directly extracted directly from the email information.

The proposed features in [22] were the most frequently used by phishing attackers, according to their sources, and in total they are 15:

- URLs Containing IP Address - binary value that determines whether there is an IP address in the URL or not.
- Disparities between HREF Attribute and link text - binary value that determines if the visible text of a link has a disparity relative to the HREF associated with the link.
- Presence of keywords in the visible text of a link- binary values that asserts if one or more keywords are in the text of a link, these words are "Link", "Click", "Here", "update" and "Login".

- Number of dots in domain name - Count the number of dots in the URL. The amount considered to be too much to be acceptable is greater than three.
- HTML Email - binary value that indicates if the email is HTML formatted, as it was found that there is a tendency for its use in phishing attacks.
- Presence of Javascript - Binary value that flags the presence of the string "javascript" either in the body of the email ro in a link, as javascript might be used by attackers to hide information.
- Number of links - number of links present in the email.
- Number of linked to domain - amount of different domains from all the urls present in the email.
- From body MatchDomain check - All domain names are extracted from the email and are compared to the domain of the sender, if there one domain does not match the email is considered phishing.
- Word list features - Specific words were found to be frequent in phishing emails, to try and catch them in the most appropriate way a division in six features was proposed, where each feature has a set of words or stemmed words. The features are a value that equals the normalized count of present keywords in the email. The word sets are the following:
  - "Update" and "Confirm".
  - "User", "Customer" and "Client".
  - "Suspend", "Restrict" and "Hold".
  - "Verify", "Account" and "Notif".
  - "Login", "Username", "Password", "Click" and "Log".
  - "SSN", "Social Security", "Secur" and "Inconvinien".

With the features normalized and ready to be used by a machine learning algorithm, they proceeded to testing their algorithm with a random forest classifier. They had available 2000 emails and they studied the impact of different dataset sizes in the accuracy of the classifier. A 10-fold Cross-validation (CV) method was used to the provide an estimate of the generalization error. The information gain was also calculated and only the eight features with the highest values were kept.

The results were very promising, but further testing should have been done, as even better results could have been obtained if other classifiers had been used, without the need for extra features.

Detection of Phishing Attacks: A Machine Learning Approach[15]

This approach tries to classify phishing emails using key features from emails and test results of different machine learning algorithms, namely Support Vector Machines (SVM, BSVM and Leave One Model Out), Neural Networks, SOMs and K-Means on

the dataset [15], to provide an overview of their efficiency in the subject of phishing detection.

The features extracted from emails were the same as the ones used by other published approaches, but some new ones were created to use the knowledge of attack methods of phishers. The sixteen features used were extracted using Python and JavaScript, and they are are the following:

- HTML Email - binary value determining if an email is HTML formatted or not, as most phishing attacks use this.
- IP-based URL - binary value determining if an email has links with an IP address instead of a domain name.
- Age of domain name - binary feature that determines if an existing domain name is less than 30 days old. A WHOIS query is done to determine this fact.
- Number of domains - number of different domain names present in the email, it is a continuous feature (not binary).
- Number of sub-domains - number of sub-domains of an URL, it is a continuous feature.
- Presence of JavaScript - binary value determining if an email has JavaScript or not, as phishers usually use this.
- Presence of form tab - binary value determining the presence or absence of forms on an HTML formatted email. Some emails might have forms asking for personal information to latter submit the information to illegitimate destinations.
- Number of links - total number of links present in the email. It is a continuous feature.
- Matching domain (From & Body) - All domain names are extracted from the email and are compared to the domain of the sender, if there one domain does not match the email is considered phishing. It is a binary feature.
- Keywords - Continuous feature that counts the number of predetermined words in an email. The words are grouped and each group is considered a feature. The groups have the following words associated:
  - "Update" and "Confirm"
  - "User", "Customer" and "Client"
  - "Suspend", "Restrict" and "Hold"
  - "Verify", "Account"
  - "Login", "Username", "Password"
  - "SSN", "Social Security"

The dataset had 4000 emails, with 973 of them being phishing emails and the others were legitimate ones. The whole of the dataset was divided, where 2000 entries were used for training the models and the remaining emails were used for testing.

The machine learning model that achieved the best results was the SVM, closely followed by the Biased Support Vector Machine (BSVM) and the Artificial Neural Networks. The algorithm that yield the worst results was K-Means clustering, making it obvious that unsupervised approaches are not as efficient.

Phishing Website Classification: A Machine Learning Approach[23]

In this approach a very thorough investigation and testing was done to try to find the best classification process, with the bare minimum features. The dataset was created with phishing URLs from PhishTank that were still active, and non-phishing URLs from Google. This was then divided into three subsets, all with a total of 1750 URLs, with different ratios of phishing URLs, 50:50, 70:30 30:70, to evaluate their performances.

The features were chosen based on research regarding this matter [24]–[26]. The features are the following, as described in [23].

- ID - Unique number of each row.
- URL - Web address of page.
- TITLE - Title of each webpage.
- HTML_Source_Code - HTML source code of webpage.
- Alexa_Rank - Value of Alexa Rank ranging.
- IP_Address - Represented as 1 if present and -1 otherwise.
- SSL_Connection - Represented as 1 if present and -1 otherwise.
- Long_URL - Length of URL.
- Dots - Number of dots present in URL reflecting how many sub-domains used.
- At_Symbol - Represented as 1 if "@" symbol is embedded or -1 if absent.
- Hexadecimal - Represented as 1 if hexadecimal codes are present and -1 if not.
- Frame - Represented as 1 if present and -1 if not.
- Redirect - Represented as 1 if the webpage has a code to redirect user to another destination and -1 if not.
- Submit - Represented 1if the webpage has a form to send data and -1 if not.
- Googe_Page_Rank - Value of Google Page Rank ranging from 0 to 10.
- Google_Position - Position of the in google search. It ranges from 0 which means non-existence to 300 hundreds.
- Label - Classification of each webpages into phishing (1) and nonphishing (0).

They evaluated several methods for detection of phishing. First, a pruning decision tree was configured and tested, followed by a number of other known machine learning algorithms. Lastly, ensembles of previously configured algorithms were tested. The models used were LR, K-NN, C4.5 decision tree and SVM. Their performance is measured by precision, recall, f1-score and accuracy.

For the pruning decision tree the C4.5 algorithm presented by Quinlan in [27]. It has the advantage of using gain ratio rather than information gain, as the last has

bias attribute selection with large number values [28], which was the case. Gain ratio changes information gain of every feature to create a more consistent scaling. It uses equation 2.1.

$$GainRatio = \frac{InformationGain}{SplitInformation} \tag{2.1}$$

Some parameters need to be defined in decision tree models. According to Akanbi et al.[23], some of these parameters are the *minimal size*, a value that is the minimum number of entries per node that if exceeded will result in the splitting of a node. The minimal leaf size is the equivalent to the previous parameter but only applies to the leaf nodes. To choose them correctly a repetitive process was done that would test the different performances of the algorithm in these different conditions.

For overfitting prevention, a post pruning process was applied, of name pessimistic pruning. This uses statistical correlation test, where an error rate is obtained and a node is pruned according to this value. For the correct use of this error an additional variable of name *confidence* is created to define the confidence level given to the error calculation. For this variable a repetitive process was done to find the optimal value.

The pruning process did not change the performance or accuracy of the algorithm, but reduced significantly the complexity and implementation time.

After completion of the decision tree tests, other classifiers are evaluated. Their procedure consisted in testing every combination of classifiers and datasets. In the case of the K-NN many values for K were tested (from 1 to 7), with the values of 1 and 2 being the best. The K-NN 1, K-NN 2, C4.5 algorithm, LR and SVM models, suffer a performance test with 9 different values for CV (from 10 to 90 where x would be 10, 20, 30, ...,90). In general, the K-NN produced the best results. The sampling method used for every test was Stratified sampling.

Having tested the performance of the models individually, they now created four ensembles to try to improve the results. The idea was to create a voting mechanic between the algorithms.

All ensembles were tested with the different datasets again, the best being the ensemble 1 (K-NN, C4.5 decision tree and LR), with a balanced dataset, meaning that they work best when the data is balanced between phishing and non-phishing entries.

Finally, after comparing the results of the best performing approaches it was found that the pruning decision tree performed the best and is closely followed by the other techniques.

2.2.5 Visual Similarity

This type of approach, according to Khonji et al. [10], has some similarities with whitelisting and blacklisting as it needs a reference dataset to find out the legitimacy

of a website, but has the advantage of being able to detect more efficiently zero-hour attacks. It relies on the assumption that phishers create websites similar to the victim's websites, hence the usefulness of comparing pages visually, as a phishing website similar to a non-phishing website that does not have the same characteristics (same domain for example) can be considered phishing. But this might reveal to be flawed as the attackers do not always copy the legitimate website.

PhishZoo: Detecting Phishing Websites By Looking at Them [29]

PhishZoo is a proposed approach for phishing detection that involves content similarity techniques. The basic idea is to create profiles of websites which are regarded as sensitive by the user, and latter use these to compare to visited websites to assert if they are trying to mimic them. It is fundamentally a whitelist method that compares visual elements between sites, and can be used in association with blacklists for detection of zero-day attacks.

PhishZoo is said to be able to detect attacks that would not be detected by URL based machine learning techniques [29]. It references also that users keep only a small set of sensitive websites with sensitive data [7], meaning that saving these is enough to protect the user against phishing attacks on these particular sites.

The workflow of this approach starts by loading a website and matching it against the stored profiles, if there is a match between SSL and URL between them the site is considered safe. If not, some features are extracted from it, namely the tokens in the hostname, URL and HTML files, and predefined keywords from the legitimate sites are compared to these. Then all images are extracted and compared to the stored ones, the images are usually logos. Finally, a website is considered phishing if there is in fact a match between these extracted features and the stored ones from the protected sites.

The image matching process relies on SIFT [30], an image-matching algorithm with Euclidean distance calculations between key points of the images. Testing reveals that almost all of the phishing websites are detected, but with false positive rate. This method is considerably demanding in processing time. Phishers try to go unnoticed by conventional image matching algorithms by changing the original image slightly, this way humans can not detect those changes, but the algorithms think it is a different image. SIFT is able to detect these small distortions successfully.

The results showed that, by only using keyword matching, the algorithm gives a better accuracy than by using both keyword and image matching, although with high false positives. It is possible to observe that a significant number of phishing sites use the same elements as the legitimate sites. Ultimately, the approach with image matching produces the best results. The major problem is that when images need to be compared, the time needed is between 7 and 17 seconds, for just one website.

Visual Similarity-based Phishing Detection without Victim Site Information [31]

The approach is able to detect phishing websites even without an initial database. Some assumption are made regarding phishing sites: that they mimic legitimate ones, that they visually similar to the targeted sites, and that phishing sites can be similar to other phishing sites. The system stores images of websites, the domain name and the correspondent label, that can be of phishing, legitimate or unknown.

The algorithm compares domains and images in order to assert if a website is phishing or if it is legitimate. A new URL is opened in a browser and both the domain and an image of the presented webpage are obtained. The image is compared to the ones previously stored. If the similarity between the images exceeds a predefined threshold, the domains of the matching images are compared as well. If the domains match too, the image is stored with the same label as the matched image. If there is an image match but no domain match, the site is considered phishing. If no stored images are matched, the result is unknown and the image is stored.

To capture a websites image a virtual screen is used and the important parts of it are extracted.

2.2.6 Hybrid approach

A hybrid approach is any approach that tries to join two or more methods described above, in an attempt to solve common issues that exist. For example PhishBlock[32] proposes a system that uses lookup lists (whitelist and blacklist) and SVM classifiers at the same time for the detection of phishing attacks.

2.2.7 Smishing attacks

In the previous sections emails were mostly used by phishers as bait. Smishing is a subtype of phishing attacks, where the communication channel utilized by phishers is through Short Message Service (SMS). Although some of the already described characteristics of phishing attacks also exist in smishing, there are a few specific peculiarities about these last that must be taken into account.

Through research it was found that there are two most used methods for detection of smishing attacks. The first are techniques based on blacklists, where any SMS sent by an entity previously recorded and categorized as malicious is blocked. The second is the extraction of information from the message and consequent data analysis by machine learning algorithms. This last one has the advantage of being able to deal with zero day attacks.

For the machine learning algorithms to be able to predict correctly the intent of an SMS they need the correct set of features that best characterize them. These features

are usually the result of a rule-based system, where it may include such rules as the one below. These rules are a collection of rules from [33], [34].

- Presence of URL.
- Content to which an existing acurl refers to. If it redirects to the download of an application or harmful site.
- Presence of mathematical symbols.
- Presence of currency signs.
- Presence of suspicious words.
- Length of message exceeds a predefined maximum.
- Presence of self-answering messages, where the sender asks the user to subscribe or unsubscribe to a product.
- Presence of characters such as numerals.
- Presence of an email address.

2.2.8 TF-IDF Algorithm

Term Frequency - Inverse Document Frequency (TF-IDF)[35] tries to find the most significant words in a collection of documents. First the so called *stop words*, such as "the", "and", are ignored. Then the algorithm computes the *Term Frequency* (TF_{ij}) for every word i in document j. TF is the ratio between the occurrences of word i in document j (f_{ij}) and the number of occurrence of the word with the highest frequency in the same document, this normalizes it giving a score from zero to one for every word in the same document. This is computed with the mathematical equation 2.2.

$$TF_{ij} = \frac{f_{ij}}{MAX_k f_{kj}} \tag{2.2}$$

Then the *Inverse Document Frequency* is calculated for every word i (IDF_i) with equation 2.3, where N is the total amount of documents in the corpus, i.e. collection of emails, and n_i the amount of documents with the term i.

$$IDF_i = log_2(N/n_i) \tag{2.3}$$

Lastly, the final scores for every word i in document j is a combinations of formulas 2.2 and 2.3, giving the final equation 2.4.

$$TF.IDF = TF_{if} \times IDF_i \tag{2.4}$$

The higher this score is, the most significant it is to characterize the corpus.

2.2.9 Performance Metrics

Out-Of-Bag (OOB) Error

This error estimation method is mostly used for decision tree based models. It uses a technique called bagging, or *bootstrap aggregation*, where random samples from the data are placed into several bags. Each tree of the decision tree based model is trained with a newly generated bag, and the error of this tree is calculated with all samples not included in that bag[36]. The OOB error is the average of all errors.

Accuracy

A performance metric meant to verify the rate of correct predictions made by a classification model. This value is calculated as shown in equation 2.5.[1]

$$Accuracy = \frac{TruePositives + TrueNegatives}{TruePositives + TrueNegatives + FalsePostives + FalseNegatives} \tag{2.5}$$

Precision

Performance metric to evaluate the rate of correct positive[1] predictions against all positive predictions. This value is calculated as shown in equation 2.6.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2.6}$$

Recall

Performance metric to evaluate the rate of correct positive[1] predictions against all truly positive samples. This value is calculated as shown in equation 2.7.

$$Precision = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2.7}$$

F1 Score

This score uses both Recall and Precision to calculate a new value that summarizes those values into one. To calculate this, equation 2.8 is used.

$$Precision = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.8}$$

[1]In the context of this thesis, positive refers to the classification of an email as phishing, and negative as the classification of an email as ordinary

Area Under the Curve (AUC)

Receiver Operating Characteristic (ROC) curves are a visual representation of the relation between true positives rate and false positive rate of the predictions made from a machine learning model. AUC is the measurement of the area bellow the line produced between the above described relation, it is the likelihood of a classifier assigning a higher score to a random positive sample than to a random negative sample.[37]

CHAPTER 3

Phishing Detection System

The phishing detection system was developed off-line, at a proof of concept level, before being implemented in the E-goi company. In this chapter is described the process of choosing the most suitable architecture in terms of number and type of the features and classification models. Different feature selection approaches and classifiers were studied. The detection systems were trained and tested with publicly available real data, i.e. emails from real users.

During the system development the experience of the E-goi collaborators was crucial. The choice of specific features was suggested based on their domain expertise.

Machine learning classification techniques were preferred instead of empirical solutions, due to their proven success in dealing with such type of problems and particularly their efficiency in catching zero day attacks.

The structure of the created concept is as follows. The first step was to understand the company's work-flow and the system requirements, as detailed in section 3.1. The next step was to obtain a suitable dataset, described in section 3.2. The process to extract the relevant information and store it on a Comma Separated Values (CSV) file for further use, detailed in section 3.3. The feature selection is described in section 3.4. The results of training and validation of phishing detection models are shown in section 3.5 with 80% of the data. The performance of the model with optimized parameters is finally evaluated with the test sub set (20 %) and the results are discussed in section 3.6.

3.1 E-goi Requirements

E-goi has two relevant types of clients for this thesis. The first are users that want to advertise a product through the company's platform. The second are people or other companies that are willing to receive these marketing campaigns. The company also

has its own specific work-flow to deal with the emails. The first type of clients create marketing campaigns in the form of emails and ask for them to be sent to a specific list of clients of the second type. Before the campaigns are sent to the people in these lists, they have to go through a series of filters. It would be at this step of the whole work-flow that the phishing detection system needs to be implemented.

The major requirement of the company was to create a system that would help to correctly filter regular emails, with the minimum amount of phishing emails classified as regular (false negative). The goal is to save time of the workers that had to go through thousands of emails to verify their authenticity, when only a small portion would actually be of phishing nature.

3.2 Email datasets

3.2.1 E-goi dataset

Originally it was intended to use a dataset from clients of E-goi. This dataset is currently under construction, it has 1200 labeled emails, where only around 30 are phishing attacks. Due to the lack of sufficiently large number of labeled phishing emails, a different solution had to be found. In section 4 a tool to solve this problem is proposed.

3.2.2 Engineered dataset

In order to create a representative and sufficiently large dataset with balanced number of regular and phishing emails more creative approach had to be applied . Major requirement was to include emails with similar format and characteristics as the type of E-goi emails (mainly marketing campaigns). This was a challenging task as the search has shown that there is not readily available dataset to comply with the requirements, it had to be somehow engineered.

The created dataset was formed from two different sources. Around 7000 emails labeled as regular were randomly selected from over 600,000 emails available in the Enron Email Dataset [38], described in more details in [39]. Around 4000 phishing emails were obtained from the Fraudulent E-mail Corpus [40] uploaded on the website [41].

All phishing emails were originally in a single text file, to be able to work with them, a separation into independent files for each email had to be done. Thus, the data were stored in files with typical *eml* extension for emails, one file per email. Within their content the emails were formated in Multipurpose Internet Mail Extensions (MIME) and were converted into HTML.

Data were separated in two sub-datasets, where 80% is used for training and validation, and 20% for testing.

3.3 Description of Feature Structures

The emails are mainly text documents and therefore the phishing detection can be seen as a text classification problem. In text classification the aim is to induce a hypothesis using an algorithm that can predict the label of new examples as accurately as possible. If a vector with one element for each occurring term in the whole collection is used to represent a document, there can be a high dimensional feature space that can bring not only computational problems, but also the over-fitting of data, which can inhibit the classifier to generalize and thus predict the classification of unseen data. As a consequence, feature selection is usually applied. They are also referred as dimensionality reduction methods since their goal is to reduce the size of the document representation, controlling the computational burden involved, whilst maintaining or improving the classification performance, as they prevent the mislead in classification. Besides their importance, it is not an easy task, as potentially useful information can then be disregarded. Feature selection aims to reduce the document representation by identifying a smaller set of terms that could represent the document more effectively.

One of the most successful and commonly used document representation is the vector space model, where a feature is defined as a word occurring in a document. Filter methods are simple methods. Some scoring measure is defined so the relative importance of a term can be represented. The score is assigned to each feature independently, then sorted according to the assigned score, and finally a predefined number of the best features is taken to form the solution feature subset. Filter methods consider attributes independently from the algorithm that will use them, relying on general characteristics of the training set to select some features and discard others.

Stopword removal can be considered a basic filter method and rely on the assumption that some words, such as articles, prepositions, and conjunctions, called stopwords, are non-informative words, and occur more frequently than informative ones. Those words are then included in a list and filtered in the document representation process, as they could mislead correlations between documents. Despite the advantages of being simple and independent of the classifier used, the above mentioned methods also have the disadvantage of totally ignoring the effect of the selected feature subset on the performance of the classifier.

Many state of the art approaches use the URL based features, where every peace of the URL is explored and publicly available black and white lists can be consulted. The developed system was to be implemented on the servers of E-goi and the algorithms had to work with what was given to them. The problem arises when most of the URLs are stripped from the emails that are fed to this server. Because of this, this thesis could not rely solely on URLs. This complicated the development process, as new approaches had

to be formulated to try and solve this problem. One solution was the use of automated keyword search, with the TF-IDF algorithm. This problem was taken in consideration when the created dataset, described in section 3.2, was being researched.

3.3.1 Feature Structure 1 - Binary Features

The state of the art analysis has shown that many of the phishing detection systems rely mostly on binary features with only some real-valued features, as seen in [22] and [15]. To try to understand the importance of this combination, and the relevance of keeping real-valued features, a binary features structure was first created.

The TF-IDF algorithm was applied to extract the most typical words used by phishers. The top 15 most frequently used words in the corpus were found to be enough for the classification purposes. This process is detailed in section 3.3.4.

The collected features are listed below, they closely follow typical binary features suggested in the literature. If the condition verifies, the feature value is set to 1, otherwise it is assigned to 0:

1. If the number of links present in the email are more than 3
2. If the URLs present in the email have different domains
3. If the average number of subdomains of the URLs is greater than one
4. If any specific form exists in the email
5. If any specific script exists in the email
6. If the email is HTML formated
7. Presence of IPs in the URLs
8. If the average number of dots in URLs exceeds 2
9. If the average length of URLs is greater than 35
10. If any "@" symbol is found in the URLs
11. If any "-" symbol is found in the URLs
12. If the client sent the email on a weekend
13. If the client sent the email outside working hours (between 8am and 6pm)
14. If the client created the account in less than a year
15. Presence of certain words in the emails. This includes the words contained in the list obtained from the TF-IDF algorithm, where every word is considered as one feature.
16. Six features that represent the presence of certain words in the emails. It includes some predefined sets of words or substrings, where each set is one feature and if at least one of the words occurs in the email the feature is set to 1:
    a) set1 - "Update" and "Confirm"
    b) set2 - "User", "Customer" and "Client"
    c) set3 - "Suspend", "Restrict" and "Hold"

d) set4 - "Verify", "Account" and "Notif"
e) set5 - "Login", "Username", "Password", "Click" and "Log"
f) set6 - "SSN", "Social Security", "Secur" and "Inconvinien"

3.3.2 Feature Structure 2 - Combination of Binary and Real Value Features

Binary features do not count the frequency of occurrences of a condition and therefore revealed to be less efficient for phishing detection as it will be demonstrated further in the thesis. For example set4 would get value of 1 both if only one word or all words of the set occur in the email. To handle this problem a new email preprocessing method was applied to extract a combination of binary and real value features that prove to be more relevant for the problem in hand.

The TF-IDF algorithm was again applied to create the list of words typically occurring in phishing emails and not frequently used in regular emails. The same process was used as in section 3.3.1.

A few features are still binary, but most of them count the number of occurrences of specific characteristics of the emails. The features are as described bellow.

1. **n_link** (Real) - Sum of the links present in the email.
2. **n_domain** (Real) - Number of different domains. Every URL present has a domain, all existing domains on a particular email are compared. The total number of different domains is the value of this feature.
3. **n_subdomain** (Real) - Average number of sub-domains. For every URL present in the email the sub-domains are counted. The final value is the average of these values.
4. **form_script_html** (Real) - Sum of three elements: any existing forms; number of existing scripts; presence of HTML.
5. **ip_at_minus** (Real) - Sum of three elements relative to URLs: presence of IP; presence of "-" or "@" symbols.
6. **n_dots** (Real) - Average number of dots in URLs present in a single email.
7. **length** (Real) - Average length of the URLs present in a single email.
8. **max_len_url** (Real) - Length of the longer URL.
9. **week_day** (Binary) - 1 if the email was sent over the weekend, otherwise is set to 0.
10. **working_hour** (Binary) - 1 if the email was sent outside working hours of the country of origin (from 8am to 6pm), otherwise is set to "0".
11. Six features that account for the presence of certain words in the email content. It includes predefined sets of words or substrings, where each set is considered as one feature. The sum of occurrences of the words in the set is the value assigned to the feature:

a) **set1** (Real) - "Update" and "Confirm".
b) **set2** (Real) - "User", "Customer" and "Client".
c) **set3** (Real) - "Suspend", "Restrict" and "Hold".
d) **set4** (Real) - "Verify", "Account" and "Notif".
e) **set5** (Real) - "Login", "Username", "Password", "Click" and "Log".
f) **set6** (Real) - "SSN", "Secur" and "Inconvinien".

12. **Kwords_name_fuzzy** (Real) - Sum of occurrences of predefined keywords similar to the client's name. These words are a combination of previously seen attacks and brand names in phishing emails. Some examples are "bank", "card", "credit" and many brand names such as "visa", "yahoo" or "wells". This keyword list is dynamic, i.e. words can be added or removed from the list. 343 words were used in this feature structure. The phisher attackers often make small changes in one or few characters of the word so that it goes unnoticed to humans and exact matching between words. Therefore, the words do not need to have an exact matching with the client's name of the analysed email, if the algorithm finds a similarity between the client's name and a keyword from the list it will count it as an occurrence.
13. **Kwords_subject** (Real) - This feature is similar to the Kwords_name_fuzzy, but now a similarity between the content of the email subject and the words from the predefined keywords list is searched.
14. **Kwords_name_exact** (Real) - similar to the Kwords_name_fuzzy12, but now exact matching between the client's name and the words from the predefined keywords list is searched. To improve the matching all letters are transformed into lower case. For example the word "Paypal" is first transformed into "paypal" and then is compared to the keyword "paypal". If all characters in the words match, the algorithm will count it as an occurrence.
15. *TF-IDF-based features* (Real) - maximum of 15 additional features are added corresponding to the sum of occurrences in the email of each of the words extracted by the TF-IDF algorithm (bag of words).

3.3.3 Feature Structure 3 - E-goi feature set

Due to the specific nature of the digital marketing emails, new features were defined to reflect the long term experience of the E-goi experts dealing manually with phishing email filtering.

Some changes in the content of the features described in section 3.3.2 were applied. Since the company deals mainly with clients from Portugal, the *TF-IDF extracted features* contains mostly Portuguese words. Some examples are "banco" and its translation "bank", "card", "dispositivo", "alerta", "desbloqueio" (unlock), "protocol", "atualizacao", "brasil" (the country Brazil). There are also many brand names.

The following new features were added to the Feature structure 2 described in section 3.3.2

1. **utc_working_hour** (Binary) - UTC working hour. This feature is based on the metadata of the sent email regarding its location and sent date and time. The *working hour* feature only takes into consideration the sender time, but an attack might be sent within working hours of the sender country but outside working hours of the company receiving the attack. This feature is set to "1" if the email was sent outside working hours in UTC time zone, otherwise is set to "0".
2. **nameVSemail** (Real) - This feature counts the matching between the client's names and its email address. It is hypothesized that the higher this value is, the lower the likelihood of the email being a phishing attack. The algorithm first separates all names of the client into a list, for example the client's name "John Mark" will create a list with two elements: "john" and "mark". Each element of this list is compared to the email address. With the same example, the client's first name "john" is compared to the email "phishing@gmail.com" where there are no similarities, but if the email is "john@gmail.com" it becomes more trustworthy. The matching of words is approximate.
3. **bad_country** (Binary) - Blacklist of countries found to be a regular source of attacks. Feature set to "1" if a country from the blacklist is detected in the email content. Some examples are the countries India, Morocco, Nigeria,Tunisia.
4. **bad_char** (Binary) - Existence of unprintable characters such as "♣", "♡" or "♭". Digits, letters, punctuation, and whitespaces are not considered as bad characters. Feature set to "1" if the email has characters considered as unprintable. These "bad characters" are usually result of strange encoding of emails. This might be an indicator of an attack, as the phishers try to hide the content of the email from the algorithms.

3.3.4 Feature Extraction Process

The database of emails was divided in two directories, one for phishing and another for ordinary emails. Using the TF-IDF algorithm, described in section 2.2.8, two distinct word lists were extracted. The words with the highest TF-IDF score, i.e. the most significant words for each class are stored. For the phishing class of emails the top 20 words were extracted, whereas for the ordinary emails the top 100 words were defined. All words present both in the phishing and the ordinary word lists were removed from the phishing word list. This way only the most characteristic words for phishing emails were kept. A maximum limit of 15 words was imposed in the algorithm, because it was concluded empirically that the TF-IDF scores for more words has a low frequency

presence in the corpus. However, this is a hyper parameter of the algorithm that needs to be adjusted when applied to different datasets.

To ensure that the extracted words are actually relevant, a fine tuning process was implemented to filter out from the list words that appear in less than 3% of the total number of emails.

The sequence of steps of the TF-IDF-based algorithm for keyword list selection is schematically presented in figure 3.1.

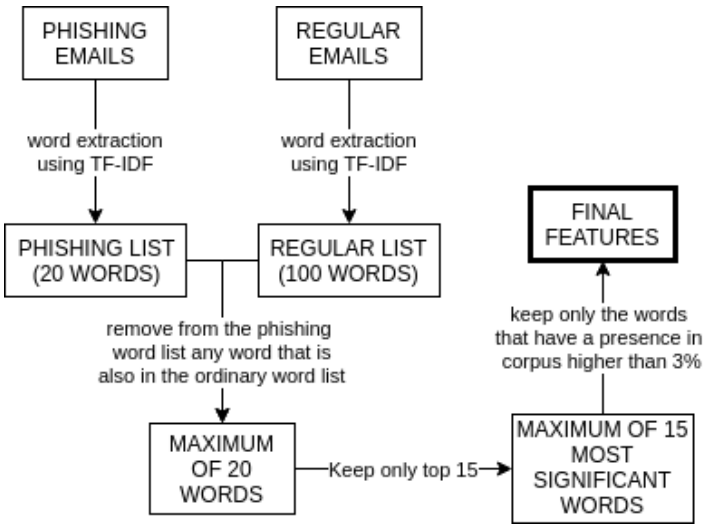

Figure 3.1: TF-IDF-based algorithm for keyword list feature selection

Each email was considered as a sample, and every sample went through an information extraction process. Some emails had HTML content, for these cases the files were opened and parsed with *Beautiful Soup*[42], a Python package for HTML parsing. All HREF tags were listed, giving all URLs contained within an email. The relevant information for each URL was extracted and temporarily saved. After this, the information was combined. First the number of links, then the number of different domains, the average number of subdomains, the existence of specific characters, average number of dots, average length and finally the length of the longest URL. Moving away from URLs, the existence of forms or scripts is asserted. If the email was not HTML formated the values of the related features were set to "0".

The next step was to parse the email with an email parser. Most emails were encoded with MIME. The Python package *email* [43] was used to parse and extract information regarding the fields of the emails, namely the date, sender details, and body (the message itself). Starting with the date, *Dateparser*[44] was used to transform the date from human readable text to *datetime* format[45], making it easier for further calculations to be applied. The features related to the date are now obtained, where the working hours are defined by the time between eight o'clock in the morning (8am)

and six o'clock in the afternoon (6pm), and the weekend is defined by Saturday and Sunday.

Finally, the content of the email is extracted. This last element contains the message sent itself, it can be in plain text or in HTML format, with or without images. A MIME decoding process is applied here.

With the content extracted, the word related features can be obtained. When the similarity between words does not need to be exact, the package *Fuzzywuzzy*[46] was used for approximate matching.

All features are stored in a list and saved in a CSV file. The file entries are then randomized and split into two distinct datasets, one for training, and another for testing. Each feature in the training dataset is normalized with a Min/Max scaler, where the minimum is zero and the maximum one. The testing dataset is normalized with the scale used to normalize the training data.

The email corpus preprocessing and feature extraction process is schematically represented in figure 3.2. The implemented code is in Python 2.7 in order to be able to use the more comprehensive library of relevant functions compared to the new version of Python 3.6. The computer system that produced the results presented in this thesis, had the Ubuntu 16.04 as operating system, 12 gigabytes of Random-Access Memory (RAM) and a Central Processing Unit (CPU) with wight cores of model Intel®Core™i7-4720HQ and clock rate of 2.60GHz.

With the computational system used the training time for models is considerably low, of at most one second, as the developed process is computationally efficient. Because of this no times are explicit in this thesis, as they could be ignored for not being a requirement imposed by E-goi nor do they present high enough values to justify any further timing optimizations.

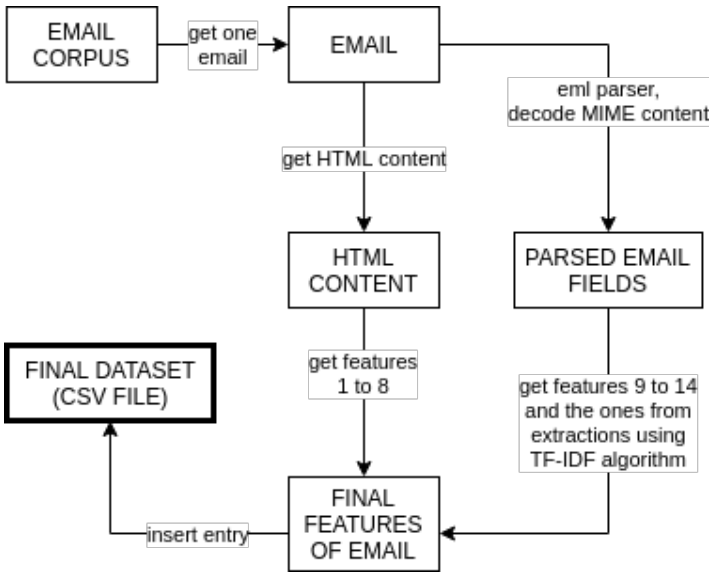

Figure 3.2: Email corpus preprocessing and feature extraction process. Feature numbers corresponds to Feature structure 2 (section 3.3.2)

3.4 Feature Visualization and Statistical Analysis

In order to evaluate the discrimination capacity of the individual features, in this section their class distribution was analysed through histogram visualization. Features from the Mixed binary and real value features, i.e Feature structure 2 described in section 3.3.2, were analyzed. Only the values from the training subset were counted in the histograms before normalization to 0-1 range. The *Seaborn* [47] library was used to for this statistical visualization. In the figures, label "0" and the green color refer to the ordinary email feature distribution, and label "1" and the red color correspond to the phishing emails feature distribution.

First, the URLs related features were analyzed. It was found that links in the emails are rare, and when they exist they are mostly from phishing emails. This fact makes the link related features relatively useful signature for phishing detection, as the classifier will find that any email with one or more links is more likely to be a phishing attack.

However, since links appear rarely in the emails, the URLs related features are not enough to solve the problem. This features are the **n_links** (figure 3.3), the **n_domain** (figure 3.4), the **n_subdomain** (figure 3.5), the **n_dots** (figure 3.6) and the **length** (figure 3.7).

The histogram visualization of the following features: the **ip_at_minus** (figure 3.8), the **form_script_html** (figure 3.9), the **Kwords_subject** (figure 3.10) and the **max_len_url** (figure 3.11), has shown that in most of the cases these features are not present in the emails (they have value 0) or have the same distribution for both classes. These results were not expected taking into account that the same features are

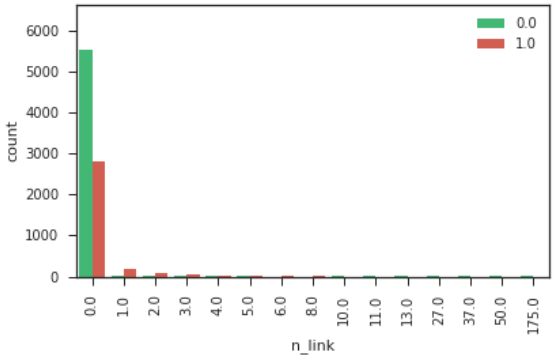

Figure 3.3: Histogram **n_links**.

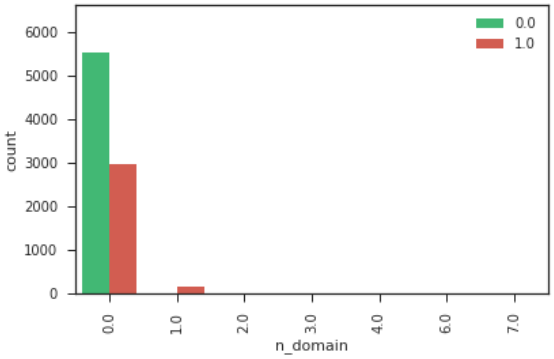

Figure 3.4: Histogram for **n_domain**.

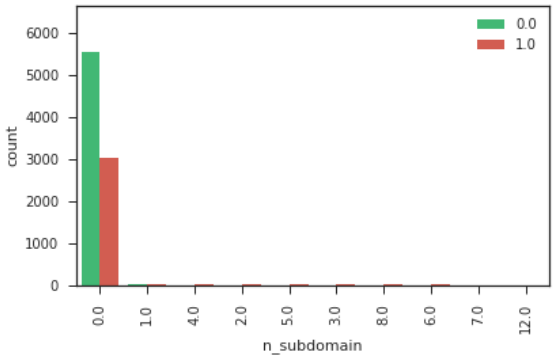

Figure 3.5: Histogram for **n_subdomain**.

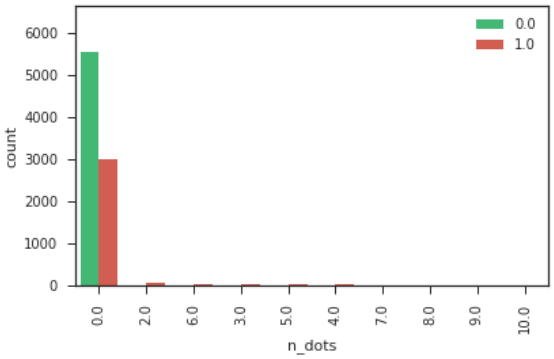

Figure 3.6: Histogram for **n_dots**.

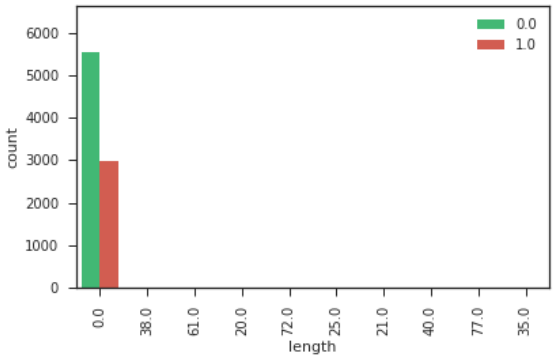

Figure 3.7: Histogram for **length**.

often suggested in the literature.

Similar to the link related features, the histograms of the **Kwords_name_fuzzy** (figure 3.12), the **set4** (figure 3.13), the **set6** (figure 3.14) and the **week_day** (figure 3.15 show clear distinction between the class distributions.

Overall, if a feature from this group has value equal or bigger than one most probably the email is a phishing one. Note that, while the link related features appear rarely in the emails, these features are typically not null in phishing class of emails. Therefore,

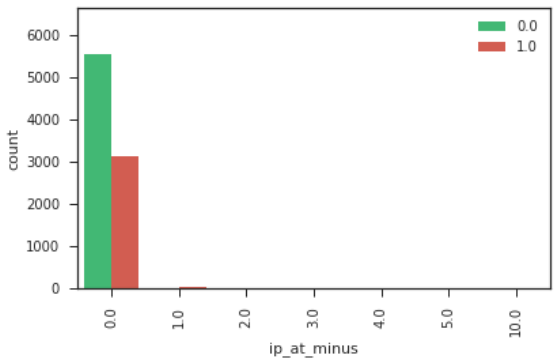

Figure 3.8: Histogram for **ip_at_minus**.

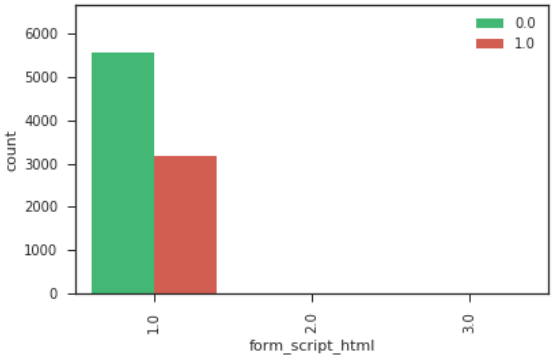

Figure 3.9: Histogram for **form_script_html**.

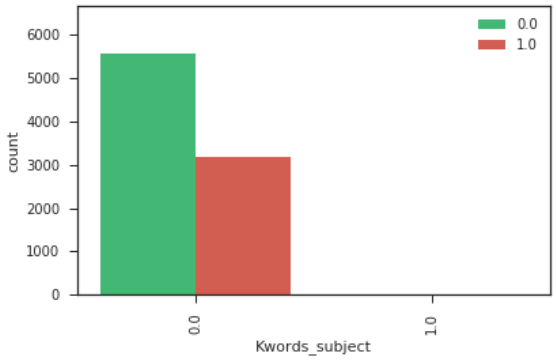

Figure 3.10: Histogram for **Kwords_subject**.

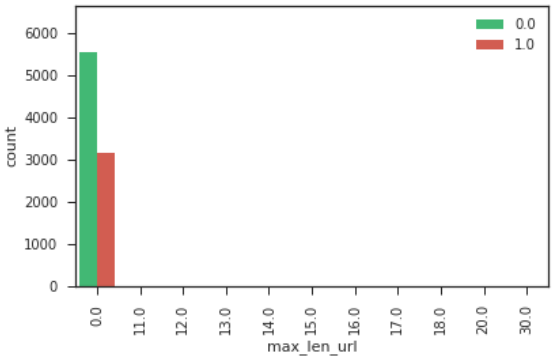

Figure 3.11: Histogram for **max_len_url**.

their discrimination capacity is stronger.

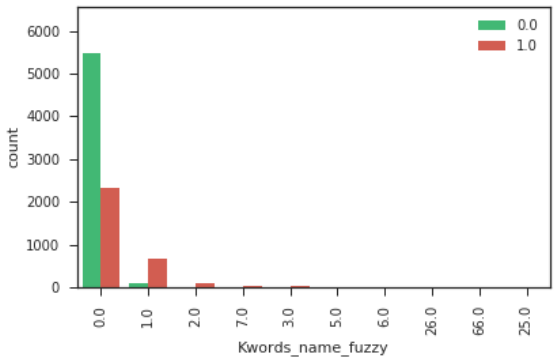

Figure 3.12: Histogram for **Kwords_name_fuzzy**.

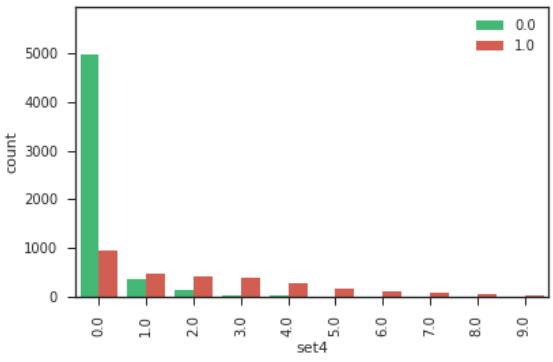

Figure 3.13: Histogram for **set4**.

In contrast to the previous group of features, the following subset, **Kwords_name_exact** (figure 3.16), **set1** (figure 3.17), **set2** (figure 3.18), **set3** (figure 3.19), **set5** (figure 3.20) and **working_hour** (figure 3.21) present ambiguous

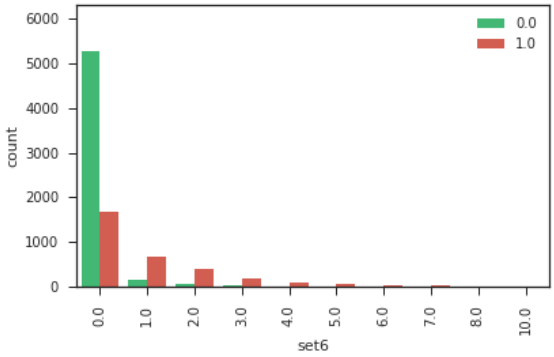

Figure 3.14: Histogram for **set6**.

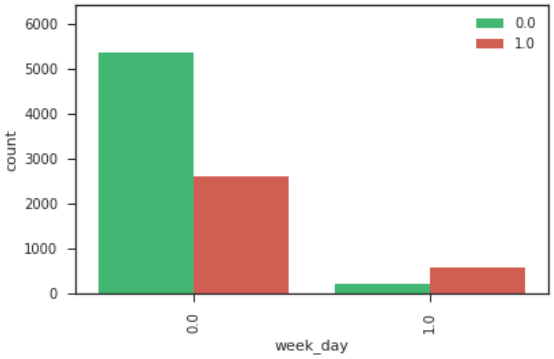

Figure 3.15: Histogram for **week_day**.

class distributions. Through observation it is not possible to make conclusions about their discrimination quality.

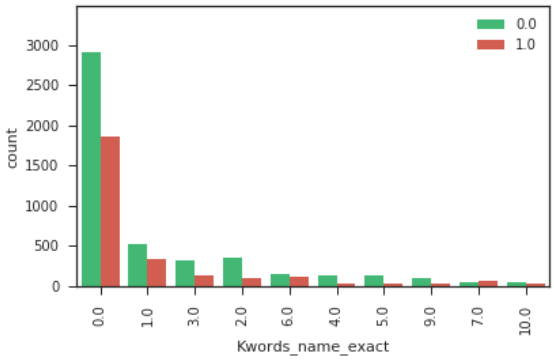

Figure 3.16: Histogram for **Kwords_name_exact**.

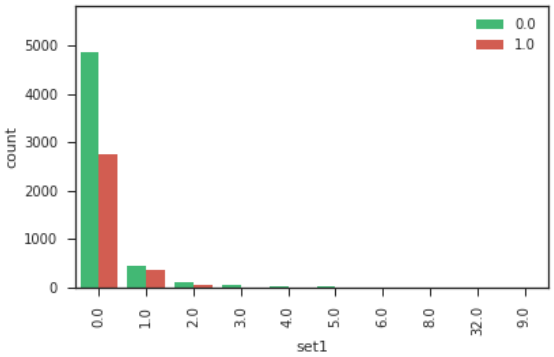

Figure 3.17: Histogram for **set1**.

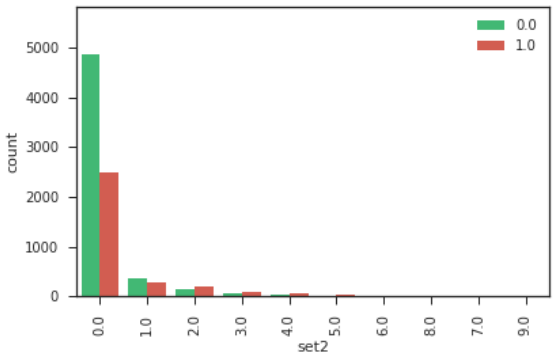

Figure 3.18: Histogram for **set2**.

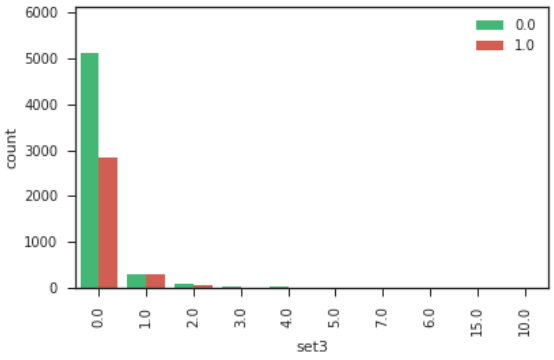

Figure 3.19: Histogram for **set3**.

The features obtained through the TF-IDF algorithm were considered separately, with an automated selection process. With the training dataset the 15 most significant words, obtained after subtraction of common significant words for both classes, were

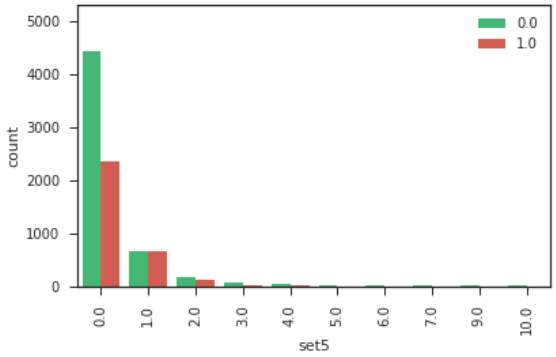

Figure 3.20: Histogram for **set5**.

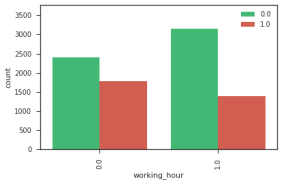

Figure 3.21: Histogram for **working_hour**.

"mainly", "meant", "media", "million", "modalities", "bank", "background", "murdered", "need", "whites", "operating", "opportunities", "options", "particularly" and "partnership". The final list of actually used features was obtained after selection of the words that appear in more that 3% of the emails. These were "bank", "media", "million", "modalities", "need" and "partnership". The class distribution of the TF-IDF related features is shown on figures from 3.22 to 3.27.

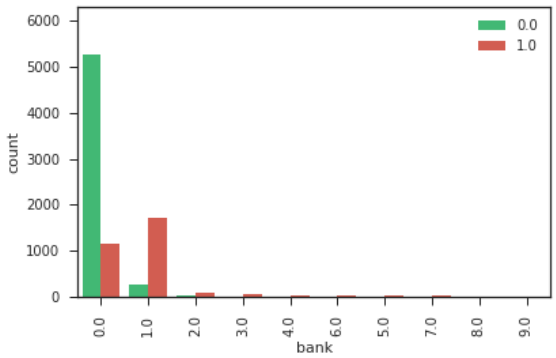

Figure 3.22: Histogram for the word **"bank"**.

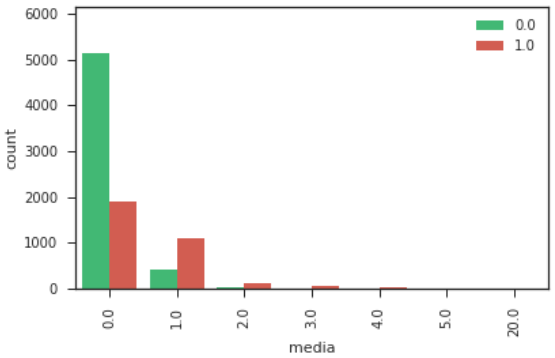

Figure 3.23: Histogram for the word **"media"**.

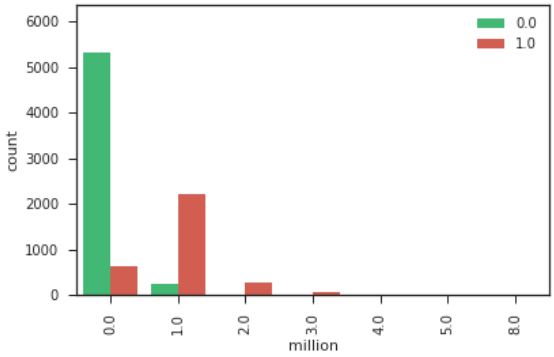

Figure 3.24: Histogram for the word **"million"**.

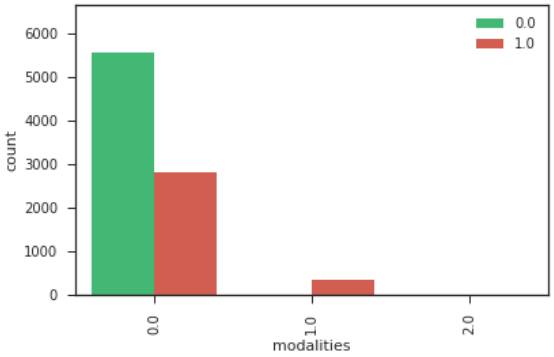

Figure 3.25: Histogram for the word **"modalities"**.

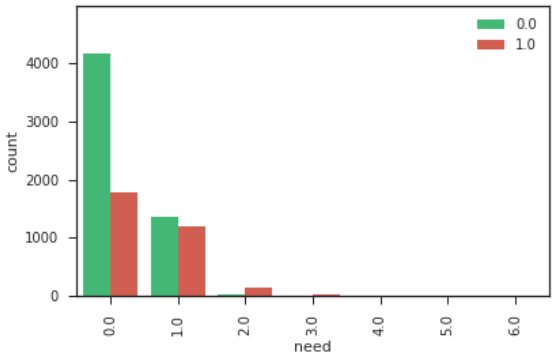

Figure 3.26: Histogram for the word **"need"**.

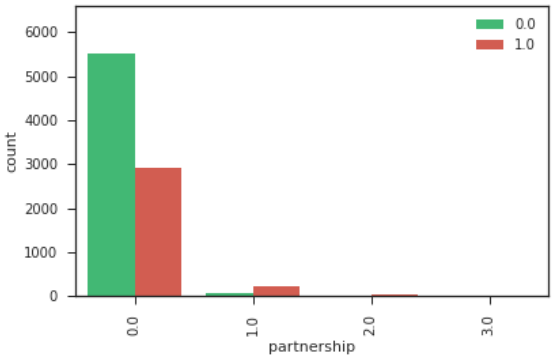

Figure 3.27: Histogram for the word **"partnership"**.

3.5 E-mail classification

The process of choosing the most appropriate e-mail classification model is detailed in section 3.5.1. Among various models comparatively studied, the Random Forest (RF) prove to have the best discrimination properties. The optimal parameters of RF model were tuned in section 3.5.2.

3.5.1 Classification Models

The classification models studied were Logistic Regression, K-NN, SVM, Decision Tree (DT), RF, and Boosted Trees. Their choice is motivated by the analysis of the state of the art. In many of the previous works tree based models were suggested as classifiers for solving related problems, K-NN and SVM are among the most widely applied machine learning classifiers in diverse applications. The optimized structure of the classifiers was obtained after tuning their most sensitive hyper parameters with training subset of the features.

The performance of the K-NN classifier in terms of Mean Square Error (MSE) was

analyzed varying the number of neighbors K in the range of 1, 50. The results are graphically represented in figure 3.28. K = 6 was determined as the optimal value.

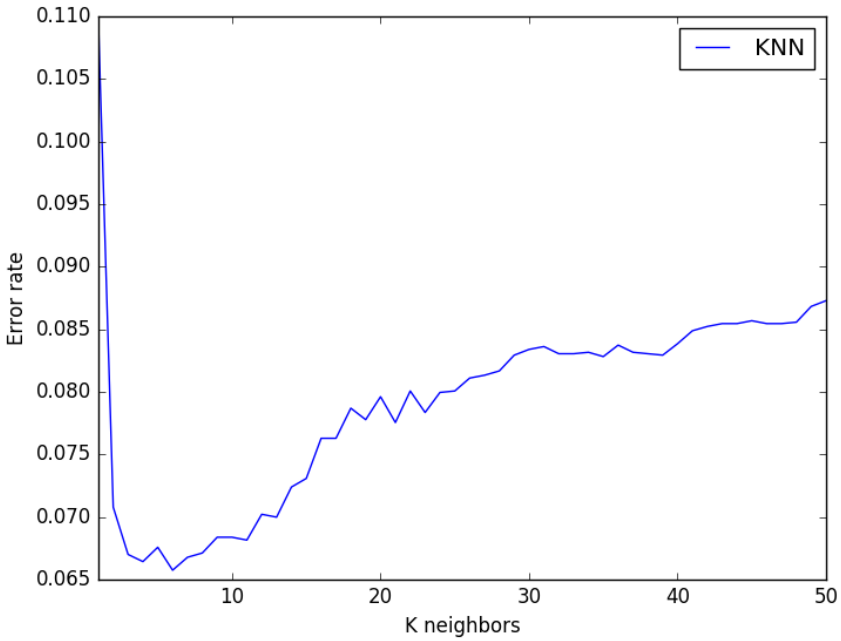

Figure 3.28: K-NN model (variation of K)

The performance of the DT model was analyzed varying the maximum depth of the tree in the range of 1 , 50. The MSE results are depicted in figure 3.29, the optimal value of the maximum depth was determined as 9.

The MSE rate of the RF classifier varying the number of estimators in the range of 1, 500 is shown in figure 3.30. The optimal number of trees in the ensemble was defined as 150.

The MSE rate of the Boosted Trees model varying the number of estimators in the range of 1, 500 is shown in figure 3.31. The optimal number of trees was found to be 80.

SVM models with Radial Basis Function (RBF), linear, sigmoid and polynomial kernels were studied. The comparative results of 5-folds cross-validation between SVM (4 models), K-NN, RF, DT and Boosted Trees classifiers are summarized in Table 3.1. For K-NN, RF, DT and Boosted Trees models the optimized configurations (discussed above) are considered. Typically the phishing detection datasets are unbalanced, normally the phishing emails are much less than the regular emails, therefore besides the accuracy, the precision, recall and F1 score are useful performance metrics. F1 score is a particularly valuable performance indicator as it underlies the percentage of correctly predicted phishing emails.

Table 3.1 shows that the SVM models with linear and RBF kernel functions are

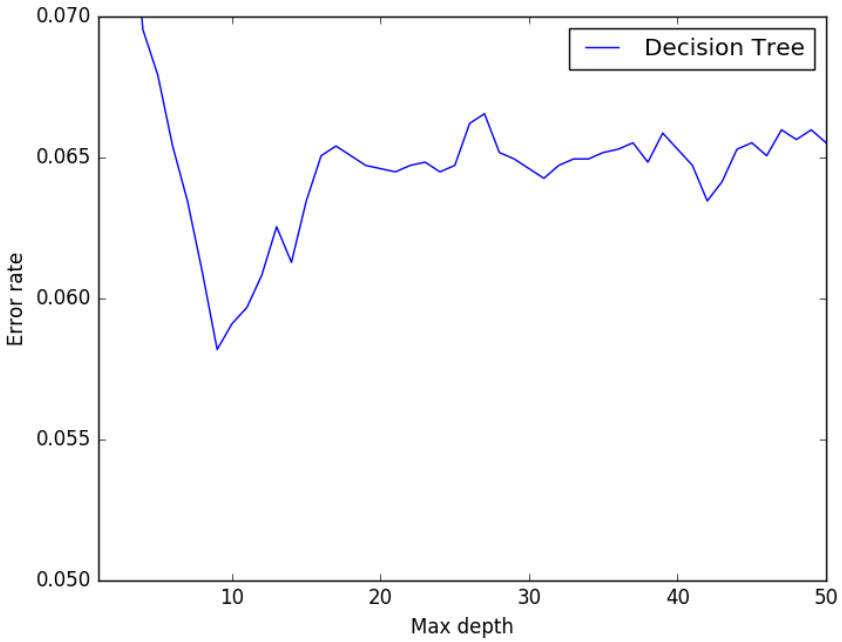

Figure 3.29: DT model (variation of maximum depth)

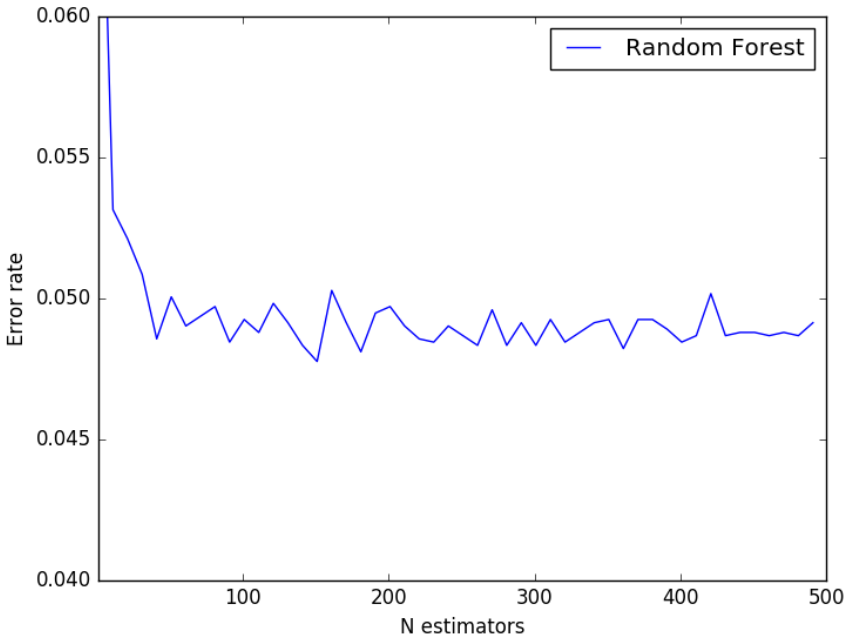

Figure 3.30: RF model (variation of maximum depth)

more reliable classifiers than SVM with polynomial and sigmoid kernels. However, the performance metrics clearly suggest that the tree based models (RF, DT and Boosted Trees) outperform the other classifiers. These results are in accordance to the reviewed literature.

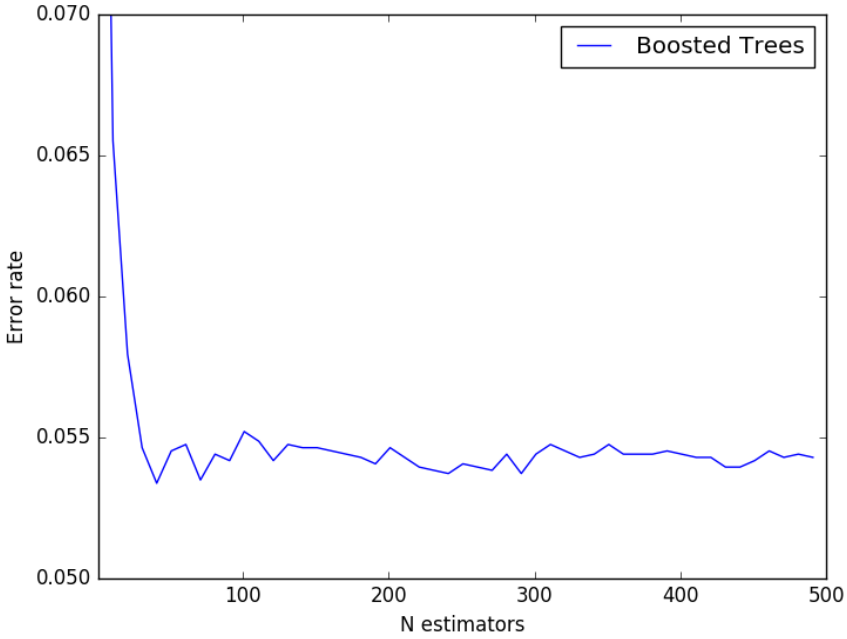

Figure 3.31: Variation of maximum depth on the training data with decision tree model.

Random Forest model is the winner in this competition, closely followed by the Boosted Trees. A potential problem with the RF model and in general with the tree based ensemble models might be the training time due to the high number of trees. However, this problem can be managed if the dataset is not extremely large, which is the case in the present work. Based on this analysis and taking into account that Boosted Trees is a more complex model to build, the RF classifier was selected to be further tuned and applied for the phishing detection task.

In addition to this study on the Feature Structure 2, and to evaluate the difference in performance between the Feature Structure 1 and Feature Structure 2, the same model configurations were trained with a dataset with features related to the Feature Structure 1. This yield somewhat worse results and proves that Feature Structure 2 is the most suited solution for this specific phishing problem, as shown in table 3.2. It is also possible to observe that the best performing algorithm was Random Forest, confirming it as the most suited classification model.

Due to the lack of sufficient phishing samples in the E-goi dataset Outlier detection approach, described in [17], was studied. The idea was to teach the model what a regular email is, and then the emails that appear to be somehow different would be considered as phishing ones. Four models were tested.

i One Class SVM, an unsupervised Outlier detection SVM, where the training data do not contain any phishing email (pure data). The other algorithms were allowed

Table 3.1: Performance indicators of 5-fold cross-validation for Feature Structure 2

	Accuracy (%)	F1 Score (%)	Precision (%)	Recall (%)
Linear Regression	91.80	88.27	92.15	84.71
SVM (Linear Kernel)	91.21	87.35	91.83	83.30
SVM (Polynomial Kernel)	66.53	17.18	87.35	9.53
SVM (RBF Kernel)	90.96	87.02	91.22	83.20
SVM (Sigmoid Kernel)	87.66	80.47	94.92	69.86
K-NN (K = 6)	93.10	90.24	93.09	87.57
Random Forest (150 Decision Trees)	**95.23**	**93.41**	**94.13**	**92.70**
Decision Tree (max depth = 9)	94.10	91.74	93.65	89.90
Boosted Trees (80 Decision Trees)	94.23	91.89	94.02	89.87

Table 3.2: Performance indicators of 5-fold cross-validation for Feature Structure 1

	Accuracy (%)	F1 Score (%)	Precision (%)	Recall (%)
Linear Regression	93.05	90.21	92.17	88.33
SVM (Linear Kernel)	92.44	89.52	89.95	89.12
SVM (Polynomial Kernel)	90.42	85.60	93.99	78.60
SVM (RBF Kernel)	93.04	90.14	92.57	87.86
SVM (Sigmoid Kernel)	91.94	88.51	91.49	85.74
K-NN (K = 6)	92.54	89.23	93.60	85.27
Random Forest (150 Decision Trees)	**93.67**	**91.14**	**92.48**	**89.85**
Decision Tree (max depth = 9)	93.40	90.69	92.81	88.68
Boosted Trees (80 Decision Trees)	93.00	90.17	91.88	88.52

to be polluted, with a parameter set to define the pollution percentile.

ii Elliptic Envelope, based on Gaussian distribution.

iii Isolation Forest, based on Random Forest algorithms, with the objective of isolating samples that diverge from regular occurrences.

iv Local Outlier Factor, based on the K-NN algorithm, where each sample is given a score according to its distance to other neighboring samples.

These algorithms were not able to reliably detect phishing emails in the E-goi dataset. They produced a high number of false negatives, considering the phishing emails as very similar to the regular ones, and therefore the Outlier approach was abandoned for further study in this thesis.

3.5.2 Building Random Forest optimal configuration

The optimal configuration of the RF model was obtained varying three major hyper parameters:

i *max_features* - maximum number of features to be considered when creating a new branch of a tree. Two cases were studied: all features and the square root of all features.

ii *n_estimators* - number of estimators, i.e. number of decision trees in the ensemble. A wide range of 1, 3000 estimators was studied.

iii *max_depth* - maximum depth that a tree can reach, i.e. the length between the leaf nodes and the root node cannot exceed this value. A range of 1, 25 tree depth was studied.

The RF models were tuned when provided with different features extracted from the Feature structure 2 (combination of binary and real value features), described in section 3.3.2. The sequence of steps of the TF-IDF-based algorithm for keyword list selection is schematically presented in figure 3.1. The final list of selected keywords is "bank", "media", "million", "modalities", "need" , "partnership".

Four sub-sets were built and considered in the experiments.

1. Feature set 1 - 25 features (complete Feature structure 2).
2. Feature set 2 - 21 features (features with lower weight were removed).
3. Feature set 3 - 10 features (features with a clear discrimination between classes were kept).
4. Feature set 4 - 19 features (TF-IDF-based features were removed).

The features that integrate each feature set are summarized in Table 3.3

The performance of the Random Forest model for different parameters that define its structure is evaluated by the Out-Of-Bag (OOB) error. OOB error is a widely used performance indicator for tree based classifiers. Each tree is trained with a bag of randomly chosen samples from the feature set and then is tested with samples not used

Table 3.3: Features considered in each feature set.

	n_link	n_domain	n_subdomain	form_script_html	ip_at_minus	n_dots	length	max_len_url	week_day	working_hour	set1	set2	set3	set4	set5	set6	Kwords_name_fuzzy	Kwords_subject	Kwords_name_exact	TF-IDF features
Feature set 1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Feature set 2	x	x	x			x	x		x	x	x	x	x	x	x	x	x		x	x
Feature set 3									x					x		x	x			x
Feature set 4	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	

for training (out-of-bag samples). The OOB error is the average of the errors of all trees with their respective out-of-bag samples.

During the experiments, first the optimal number of trees was determined (*n_estimators*), which is the most important parameter of the RF model. Next, the optimal *max_depth* parameter was determined through experiments where the RF model has the optimal number of trees defined at the previous step.

The results of the OOB errors are graphically presented where on each plot are depicted two curves corresponding to the two cases of the *max_features* parameter, namely all features and sqrt(all features). For example from figure 3.32, one can conclude that the optimal configuration of the RF model is with 400 trees and with maximum of 5 features to be considered when creating a new branch of a tree. The OOB error for this configuration is 4.7%. The model was trained with the complete feature set of 25 mixed binary and real value features. Adding more estimators increases the OOB error that suggests overfitting problem.

In the next experiment we tuned the *max_depth* parameter for a RF model with 400 trees (defined at the previous step). The results are summarized in figure 3.33. Combining now the results of the two experiments we can define the final structure of the RF model as 400 trees, 5 *max_features* and maximum depth of 20. The OOB error of this model is 4.6%.

The next set of experiments were performed with Feature set 2 (21 features), where features with less importance were removed. The removed features are those that in the statistical analysis based on histogram visualization revealed to have overlapping class distribution (see section 3.4).

The same approach was followed, as with the Feature Set 1, tuning first the number of estimators and then the *max_depth* parameter. Taking into account the dimension of the feature set the values of the *max_features* parameter are now set to 21 and 5.

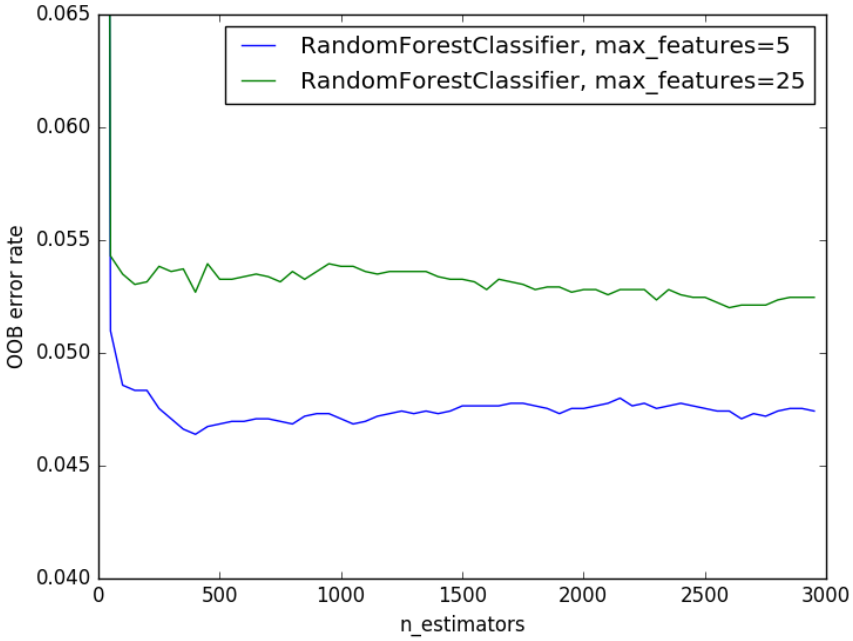

Figure 3.32: OOB error for Feature set 1, varying *n_estimators* parameter

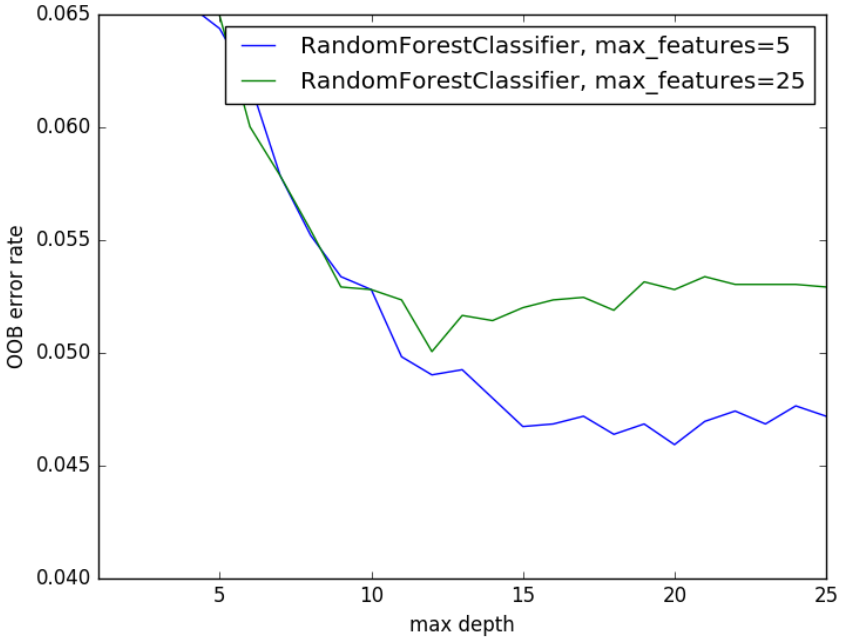

Figure 3.33: OOB error for Feature set 1, varying *max_depth* parameter

The results are plotted in figure 3.34 and figure 3.35. The optimal configuration of the RF classifier was defined as 900 trees, maximum depth of 16 and *max_features* of 5. The OOB error of this model is 4.4%. Increasing the number of trees did not bring

advantages. The experiments with Feature set 2 confirmed that the excluded features does not contribute indeed to the class discrimination. On the contrary, a slightly lower OOB error was achieved with the reduced feature set.

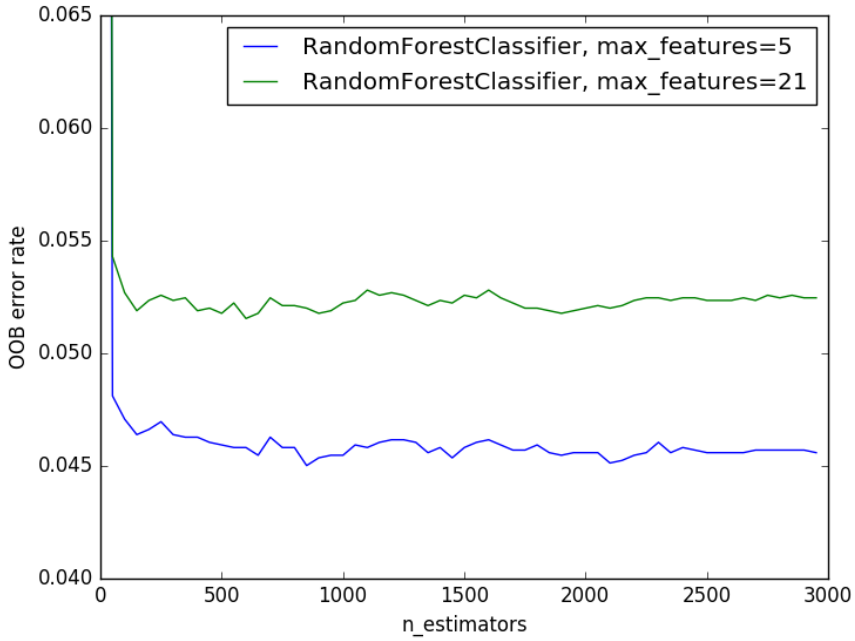

Figure 3.34: OOB error for Feature set 2, varying *n_estimators* parameter

The third set of experiments were performed with Feature set 3 (10 features), where only the features found to have a clear discrimination between classes were kept. All link related features (typically suggested in the literature) and word sets (set1, set2, set3, set5) considered as less important during the statistical analysis were removed. The results are plotted in figure 3.36.

For both values (3 and 10) of the *max_features* the OOB errors are higher than with the previously used feature sets. The minimum error achieved was 6.5%. These results prove that Feature set 3 lacks information and is therefore less appropriate for phishing detection problem. No further studies over this set were done.

Similar results were obtained when the RF model was trained with Feature set 4, where the TF-IDF-based features were removed. Figure 3.37 shows that this is the worst set of features, with minimum OOB errors of 8.8%. No further studies over this set were done.

The performance achieved by the optimal RF configurations obtained by the four set of experiments are summarized in Table 3.4.

After careful consideration, it is possible to assert that Feature set 2 provides the best mixture of binary and real value features combining email meta data, such as date

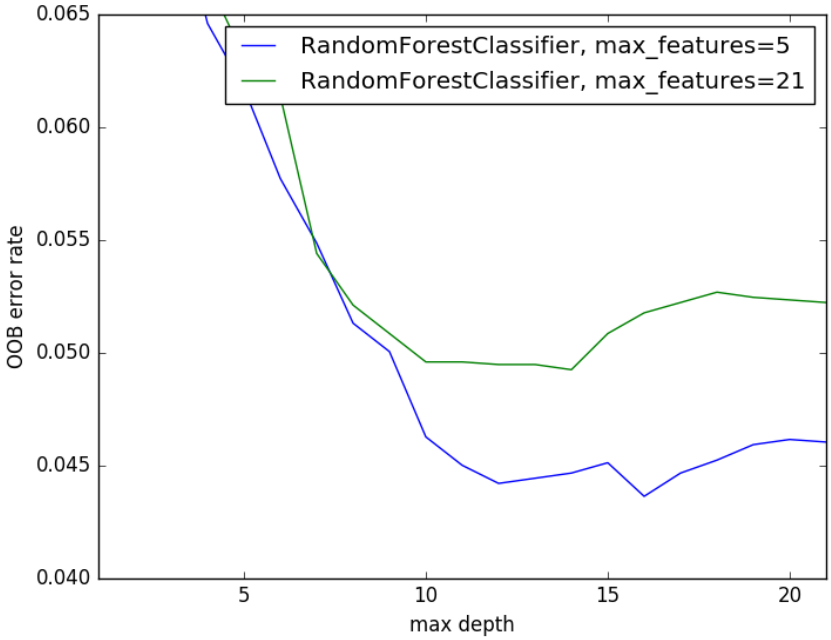

Figure 3.35: OOB error for Feature set 2, varying *max_depth* parameter

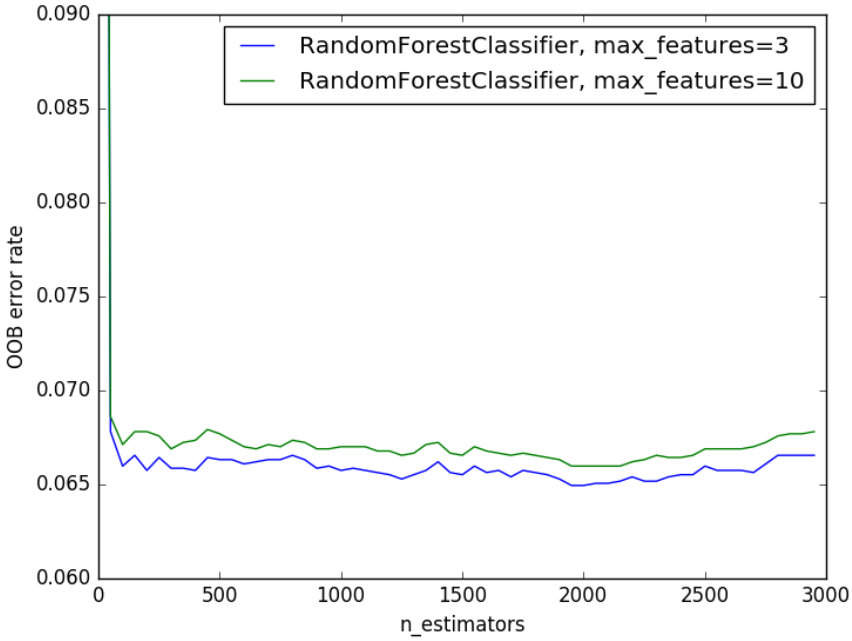

Figure 3.36: OOB error for Feature set 3, varying *n_estimators* parameter

and links, and bag of words data extracted from the email content.

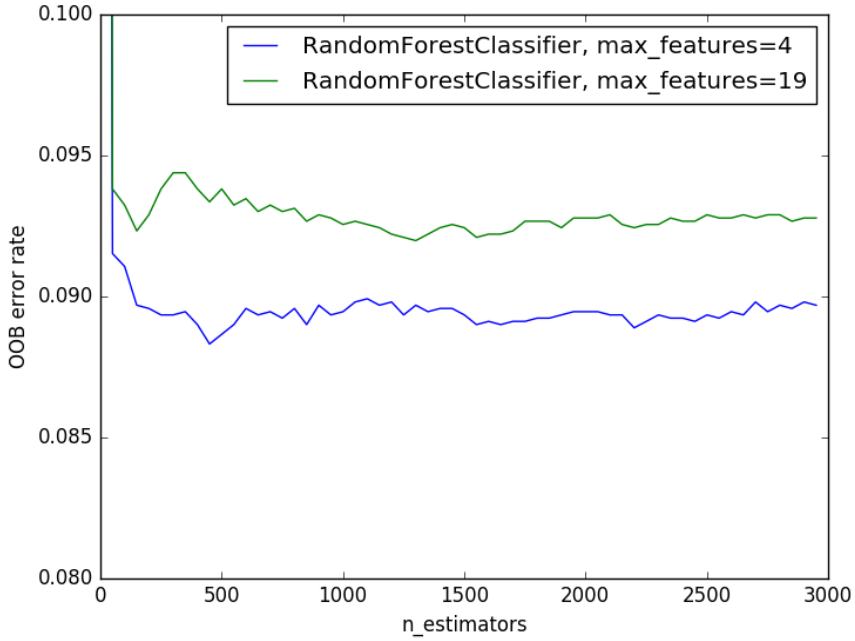

Figure 3.37: OOB error for Feature set 4, varying *n_estimators* parameter

Table 3.4: Random Forest optimal parameters and performance results (OOB error) of Configurations 1 and 2.

	Configuration 1	Configuration 2	Configuration 3	Configuration 4
Feature Set	Feature Set 1	Feature Set 2	Feature Set 3	FeatureSet 4
n_estimators	400	900	2000	400
max_depth	20	16	No Limit	No Limit
max_features	5	5	3	4
OOB error	4.6%	4.4%	6.5%	8.8%

3.6 Random Forrest generalization performance

After the exhaustive search for the most appropriate feature set and the optimal classifier detailed in the previous sections, in this section the generalization capacity of Configuration 1 and Configuration 2 models is studied. The two RF models are faced with the test set of emails not used in the previous experiments. 20% from the total email data were randomly chosen and stored apart at the beginning of the project. The results are presented in Tables 3.5, 3.6 and 3.7.

Note that the performance indicator Accuracy is higher than the F1 score. This is an expected result due to the availability of more regular email samples than phishing ones (unbalanced data). This suggests that the classifier learned to recognize better what a regular email is than a phishing email, therefore the F1 score is lower. The

Table 3.5: Generalization performance indicators for Configuration 1 and 2

	Configuration 1	Configuration 2
Accuracy (%)	95.37	98.63
F1 Score (%)	93.24	97.99
Precision (%)	94.57	99.32
Recall (%)	91.95	96.70

Table 3.6: Confusion Matrix of Configuration 1 with test data

		Predicted Outcome	
		Phishing	Regular
Actual Value	Phishing	TP: 697	FN: 61
	Regular	FP: 40	TN: 1385

Table 3.7: Confusion Matrix of Configuration 2 with test data

		Predicted Outcome	
		Phishing	Regular
Actual Value	Phishing	TP: 733	FN: 25
	Regular	FP: 5	TN: 1420

Configuration 2 outperforms Configuration1 with respect to all performance metrics (accuracy, F1 score, precision, recall). Additionally, figures 3.38 and 3.39 (zoomed version for better visualization) show the ROC curves regarding the two configurations.

Based on all performance results we can safely recommend Configuration 2 as the most trustful classifier, among the ones studied, with good generalization properties regarding both classes.

The final experiment aims to study the effect of the number of training examples on the classifier accuracy. The results are plotted in figure 3.40.

As it is expected, more training examples will decrease the gap between training and cross-validation accuracy. Getting more training examples may worsen the classifier accuracy on the training data however it will improve its generalization properties.

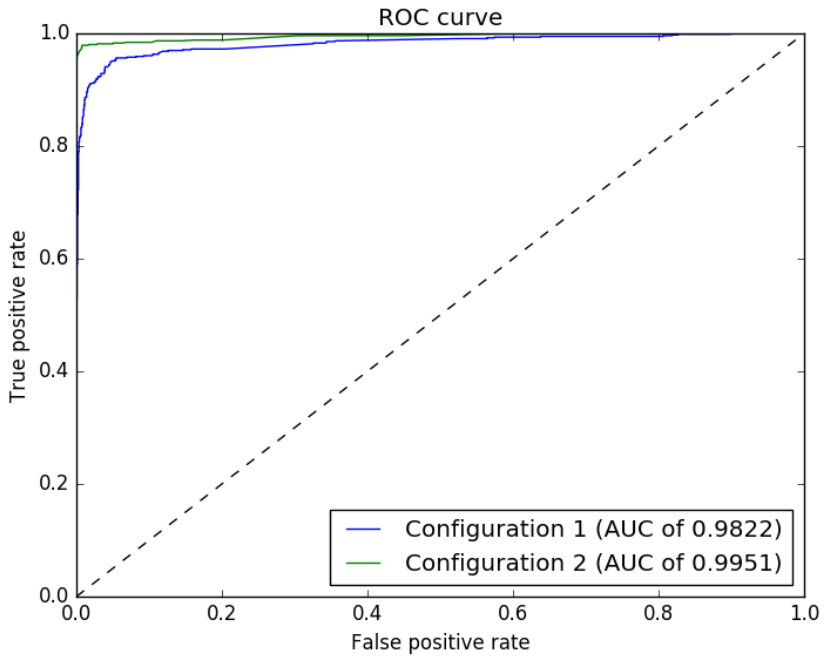

Figure 3.38: ROC curve for Configuration 1 and Configuration 2.

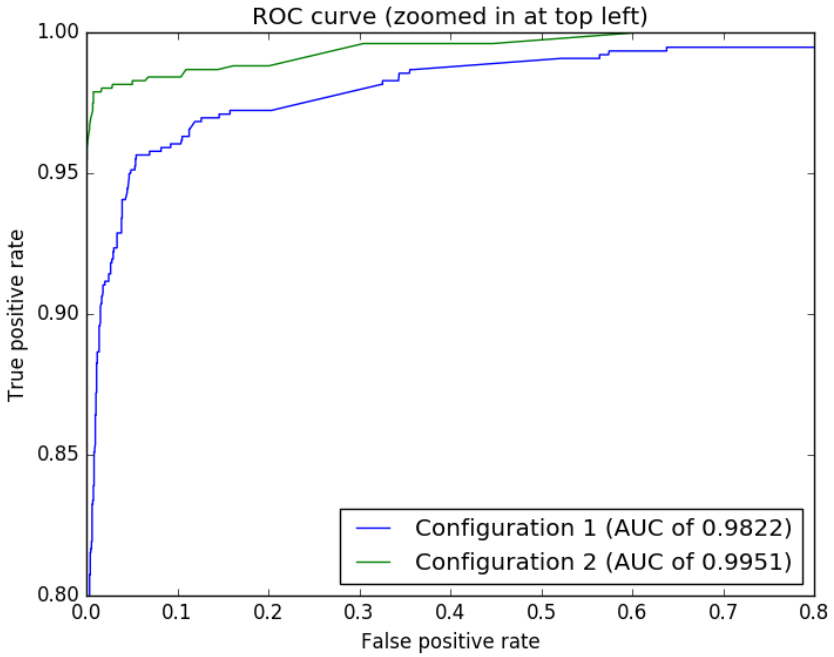

Figure 3.39: Zoomed ROC curve for Configuration 1 and Configuration 2

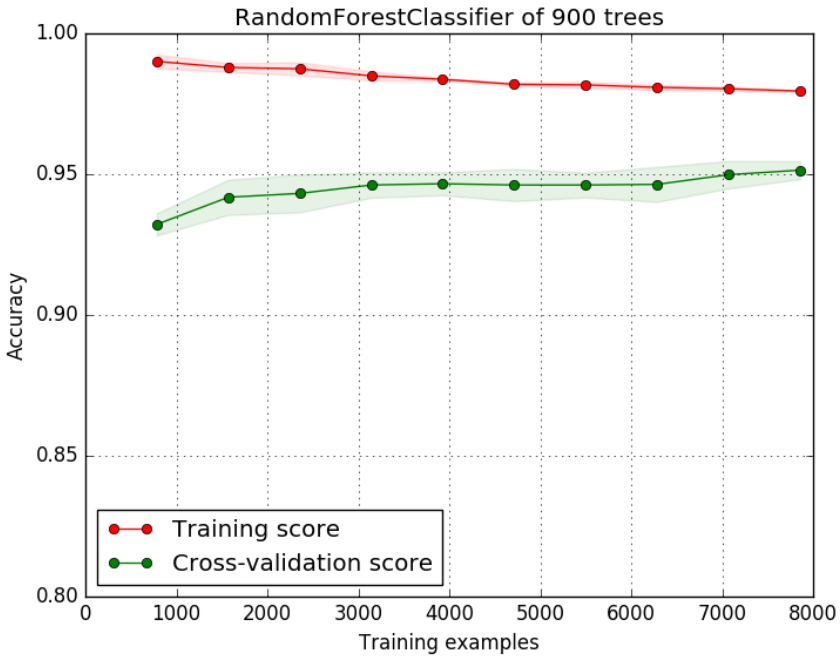

Figure 3.40: Learning curve of RF model.

CHAPTER 4

Email Labeling Web Tool

It was agreed with E-goi that there was a need for the collection of data, and consecutive labeling. For this purpose a web tool was developed. Its objective was to create an easy to use platform to label the marketing campaigns in the form of emails, so that a more conclusive study on the data can be in the future.

The developed tool is meant to be used exclusively used by workers of E-goi with enough experience to correctly differentiate phishing emails from regular ones.

In order for this tool to function correctly, an external service must be feeding the system with additional unlabeled emails, when available. In the systems of E-goi the incoming advertising emails are processed and sent to different services, it is at this point of the processing of emails that they should be copied into the assigned directory for unlabeled emails.

4.1 Back End

This tool runs with a Python microframework with the name of Flask [48] and creates the actual web pages with simple HTML files, where some JavaScript was applied.

This tool has a feature that offers an advice on the classification of emails. This is possible with a machine learning model that is trained on the background. Users might use this features to help them classify unlabeled emails.

Upon starting the service, if a model is not already available for classifying the data, a new model is trained with the already labeled data. This trained model is not supposed to be final, it only has representative value. It gives a rough estimate of how the model will react to the available data.

The algorithm uses whatever data is labeled to train the models, going up to a maximum of 4000 emails. The sub set of emails used is created automatically and is always balanced, meaning that the amount of phishing emails is always the same as

the regular emails. This is to prevent the cases where the dataset would become too unbalanced.

From each email, the extracted features are the ones described in section 3.3.3. Then, the data is fed to a grid-search algorithm from Scikit-learn [17], where the best parameters of a random forest classifier are exhaustively searched through a grid of values. This grid consists of a set of predefined values for each parameter to be tested. The machine learning model used was Random Forest and the parameters that were tested with the grid search were the maximum depth of the trees (of values 5,10,20,30), the number of estimators (of values 35,50,75,100) and the minimum samples a leaf node can have (of values 10, 15, 20). It is worth mention that during development of this advising feature for the web tool the conclusions and results obtained from this thesis had not been yet obtained, as this tool had to be ready and working on a considerably earlier date, so the parameters and their values are somewhat different from the work described in chapter 3.

Some dimensionality reduction algorithms are also tested, such as Principal Component Analysis (PCA) and SelectKBest. PCA is an algorithm that tries transform a set of variables thought to be related into a smaller set of variables, reducing the complexity of the information [49]. PCA reduced the total number of features to 4, 9, 14, 19, 24. SelectKBest algorithm selects only the K features with the most weight, where the wight is measured through a score function (which was the Chi-squared distribution), and K is 10.

Every combination of parameters in the grid search is validated with cross validation of 5 folds. In the end, the combination of dimensionality reduction algorithms and machine learning models that yield the best scores would actually be used for predictions. This training process can be triggered by users with the button "Update Classifier" on the "AI Management" page.

Emails are files with the Electronic Mail (EML) extension and are located in a specific directory. All unlabeled emails are placed in an dedicated directory. When a label is assigned to an email, it is its correspondent file is moved to the directory that contains all emails with the same label. The labels can be of *Phishing*, meaning that the email is malicious, and can be *OK*, implying they are harmless. An additional option to *Ignored* exists for the cases where none of the previous labels accurately describes the email, or if the email is not correctly formated.

The tool can handle several users classifying emails. Each user, with their own browser, is given a random email to classify. If, by chance, one email is being classified by more than one user, the only recorded answer is the first to be submitted.

Figure 4.1 is a diagram that shows how the mails are processed.

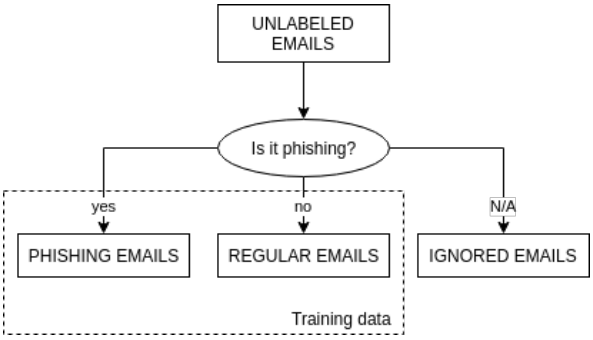

Figure 4.1: Dataset creation.

4.2 Front End

This tool consists in a few web pages. Namely the "Label-Ishing" page (which is the home page), one for labeling the data named "Classify Emails", one for evaluating results and managing the model named "AI Management", and a last one when there are no more emails to classify.

The first page that shows when accessing this tool is the home page, shown in figure 4.2. A simplistic design was chosen, where a top bar is always present for easy navigation. Users can switch between the home page, the email classification page and the machine learning and results page with a simple press of a button.

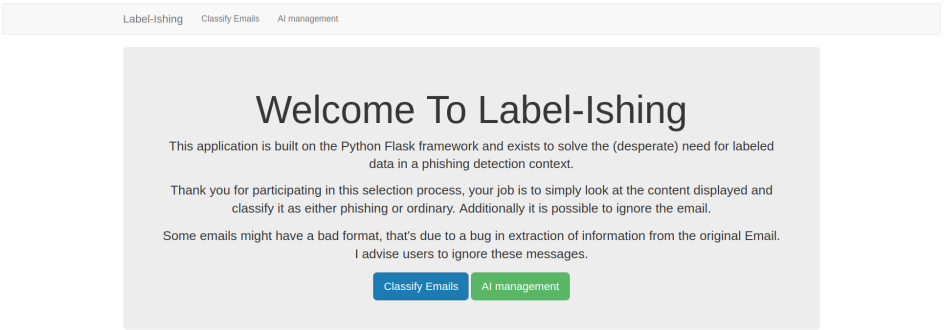

Figure 4.2: Web tool home page.

Upon pressing the "Classify emails" button, the user is redirected to the classification page, shown in figure 4.3. In this page the user is shown one random unlabeled email, and four different actions can be performed.

- Show images - Some emails contain images that are loaded from sources. To prevent cases where the attackers might be tracking whoever loads them, all

images are stripped from unlabeled emails. If they are actually necessary for the classification of an email, the user can press the "Show Images" button to load them.

- Phishing email - when a user identifies an email as being phishing, the button "Phishing" should be pressed to assign that label to it.
- Ordinary email - when a user identifies an email as regular, or ordinary, the button "Ordinary" should be pressed to assign that label to it.
- Ignore email - if the email does not fall into any previous category, or if it has a bad format or is in a strange language, the user has the option to ignore the email by simply pressing the "Ignore" button.

The figure also shows how the information regarding the emails is presented. The email used to show the functionalities is not from clients of E-goi, as no authorization was given for this purpose. This disclaimer is only referring to this representation, the working website uses real data.

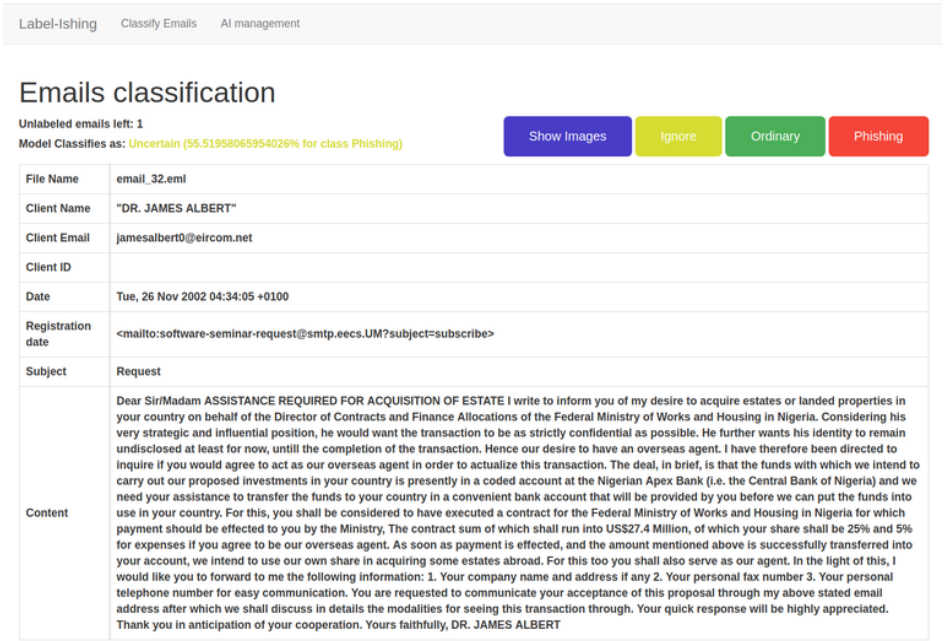

Figure 4.3: Web tool webpage for labeling emails.

Users presented with the email classification page have four distinct buttons available. The first, "Show Images", has the purpose of showing images. As default, no images contained within emails are shown, as they are usually obtained through access to the sender website. As explained above, if a shown email is of phishing and it does contains an image, it is not advisable to load them, as the attacker's website might be able to record some information of the machine that accessed it. This button allows users to load the images at their own risk.

The three other buttons, the "Ignore", "Ordinary" and "Phishing", have the purpose of labeling the data. Upon pressing one of these, the file correspondent to the shown email is moved from the *unlabeled emails* directory to directory correspondent to the label assigned to it, as described in section 4.1.

Some additional information is also available. This includes a label showing the amount of unlabeled emails, and another label that shows how the currently trained machine learning model classifies the email that is presented to the user, as well as its certainty. This last label is the content described by the text "Model classifies as:" in figure 4.3.

Lastly, the relevant information that is extracted from the email file is shown to the user in a table. This information includes the name of the file itself, some details about the client of E-goi that sent the email, such as the name, email, ID and client registration date, the send date, the subject and the content itself.

Figure 4.4 shows the "AI Management" webpage for machine learning model management and results visualization. On the left of the figure, the confusion matrix and some score metrics are shown. These results are relative to how the active prediction model predicts the label of an email and the actual label given to the same email. The values presented are the accumulation of all labeled emails since the last model update. When the "Update Classifier" button is pressed, the classifier is retrained and all score metrics are reset to zero. On the right of the figure, a count of the emails in each correspondent directory is presented, giving a rough estimate of the labeling progress up to that point.

Also, the "Set Threshold" button updates the certainty threshold value to which the classifier will consider itself in doubt. As previously explained, there is a feature that automatically gives an advice on the class of unlabeled emails, this advice consists on the certainty percentile of the prediction model for the most probable outcome. The threshold is an assigned value that defines the certainty of the model, if the predicted probability for a class is higher than the threshold then the certainty is considered high, but if it is lower, the prediction is "uncertain".

When the user tries to classify a new email and there are no more emails to classify, the user is redirected to a page informing the user about this fact. This page is shown in figure 4.5.

In figure 4.6 a diagram demonstrating the possible actions with the developed tool is visible. This describes the processes executed upon the start of the application and the processes triggered by users when using it.

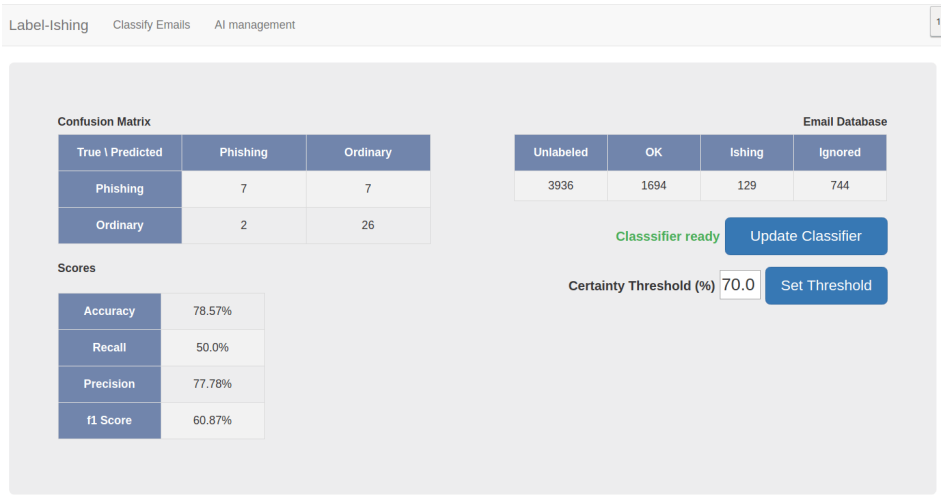

Figure 4.4: Web tool webpage for machine learning model management and results visualization.

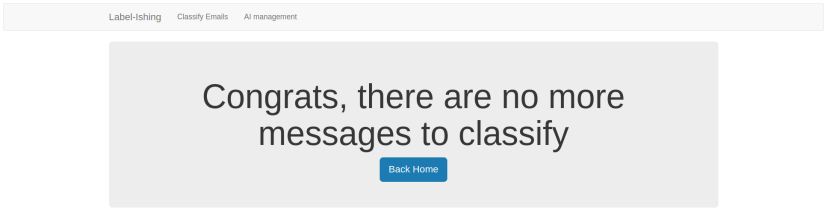

Figure 4.5: Web tool webpage for when there are no more emails to classify.

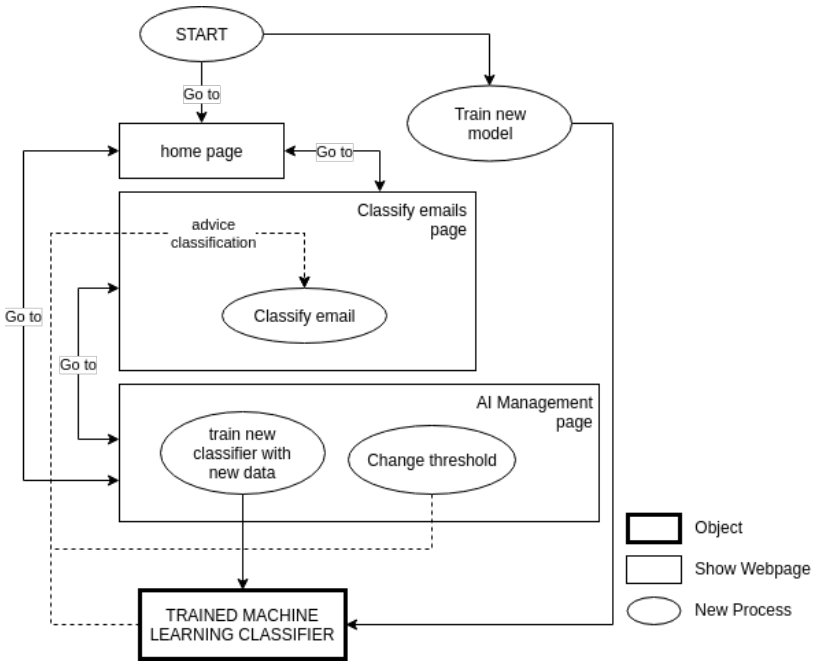

Figure 4.6: Diagram of executable processes.

CHAPTER 5

Conclusions and Future Work

The goal of this master thesis was to develop an intelligent tool to filter out the phishing emails based on machine learning methods, for the Portuguese multichannel marketing automation company E-goi. On a regular day, E-goi receives hundreds of marketing email campaigns and redirects them to a list of clients. Some of these campaigns are attacks that intend to steal user information from the end clients such as bank accounts, private data, personal logins and identity, etc. The company personnel need to go through all emails to find and block these attacks. This is a very time demanding and not scalable way to do the job. The proposed system aims to reduce the workload by automatically flagging phishing emails.

A relevant dataset, including around 7000 regular and 4000 phishing emails, was created from two online available email data sets containing emails from the two types: phishing and ordinary.

The first contribution of this work was the selection of the most appropriate features to discriminate between ordinary and phishing emails. A number of features were extracted from the emails, such as related to the links and the correspondent URLs, the presence of HTML, scripts or form in the email body, the send date and hour, counting specific word presence in the email, matching words between the client info and email subject. Additionally, keywords (maximum of 15) matching the email body content are obtained applying the text mining algorithm of name TF-IDF.

The second contribution is the proposed mechanism for phishing email detection. Most state of the art phishing detection systems are based on URL analyze and the use of outside resources, such as blacklists, whitelists or other Application Programming Interfaces (APIs). The proposed configuration integrates the full content of the email in the classification process, and is able to achieve competing results working only locally, i.e. offline, mainly due to the use of data mining techniques and the automatized

extraction of features. The system is able to dynamically find the most relevant words (features) from the email corpus and quickly adapt to new trends of phishing attacks.

After a comprehensive study of the relevance of different combinations of features, two winning configurations were defined. Both of them use Random Forest ensemble classifiers. The optimal configuration evolves 21 features, consists of 900 decision trees, has a maximum tree depth of 16, and has a maximum number of features to be considered when creation a new branch of the tree equal to 5. The performance of the classification system achieved accuracy of 98.6% and F1 score of 98.0% with test data, which is a very promising state of the art result.

The third contribution of the present thesis is the developed web tool for email labeling. The web tool allows authorized personal of the company to label the emails obtained from their real clients. This tool is expected to facilitate the implementation of the phishing detection prototype in the company. It will feed continuously the dataset and the classifier with new phishing examples, which was the major obstacle for not being possible an immediate release of the developed prototype.

Future work will also involve retraining the classification model with dataset in Portuguese and Spanish as they are the most frequently used languages within the company's clients. Currently the tool suggests training based on the whole accumulated past data (which is still rather small), however at some point it will be necessary to limit the past information only to some recent past window of emails. This will guarantee the smooth adaptation to new attack trends.

References

[1] L. Criddle, "What is social engineering?", [Online]. Available: https://www.webroot.com/au/en/home/resources/tips/online-shopping-banking/secure-what-is-social-engineering.

[2] J. Shi, S. Saleem, M. Gibbens, N. Maheshwari, P. Mueller, B. Yadegari, M. Ward, K. Fligg, G. Max, S. Martin, M. Tokutomi, E. Debusschere, and M. Mccambridge, "Csc 566, computer security research reports", no. April, 2012.

[3] I. Crime, "Internet crime report - 2010", p. 24, 2010.

[4] *Google safe browsing*. [Online]. Available: https://www.google.com/safebrowsing/static/faq.html.

[5] P. Prakash, M. Kumar, R. Rao Kompella, and M. Gupta, "Phishnet: predictive blacklisting to detect phishing attacks", *Proceedings - IEEE INFOCOM*, 2010, ISSN: 0743166X. DOI: 10.1109/INFCOM.2010.5462216.

[6] M. Sharifi and S. H. Siadati, "A phishing sites blacklist generator", *AICCSA 08 - 6th IEEE/ACS International Conference on Computer Systems and Applications*, pp. 840–843, 2008. DOI: 10.1109/AICCSA.2008.4493625.

[7] Y. Cao, W. Han, and Y. Le, "Anti-phishing based on automated individual white-list", *Proceedings of the 4th ACM workshop on Digital identity management - DIM '08*, p. 51, 2008, ISSN: 15437221. DOI: 10.1145/1456424.1456434. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1456424.1456434.

[8] D. Florencio and C. Herley, "A large-scale study of web password habits", *Proceedings of the 16th international conference on World Wide Web - WWW '07*, p. 657, 2007, ISSN: 08963207. DOI: 10.1145/1242572.1242661. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1242572.1242661.

[9] B. B. Gupta, N. A. G. Arachchilage, and K. E. Psannis, "Defending against phishing attacks: taxonomy of methods, current issues and future directions", *Telecommunication Systems*, pp. 1–32, 2017, ISSN: 1018-4864. DOI: 10.1007/s11235-017-0334-z. [Online]. Available: http://link.springer.com/10.1007/s11235-017-0334-z.

[10] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: a literature survey", *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 2091–2121, 2013, ISSN: 1553877X. DOI: 10.1109/SURV.2013.032213.00009.

[11] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell, "Client-side defense against web-based identity theft", pp. 1–16, 2004. DOI: 10.1.1.65.679.

[12] D. L. Cook, V. K. Gurbani, and M. Daniluk, "Phishwish: a stateless phishing filter using minimal rules", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5143 LNCS, pp. 182–186, 2008, ISSN: 03029743. DOI: 10.1007/978-3-540-85230-8_15.

[13] Y. Zhang, J. Hong, and L. Cranor, "Cantina: a content-based approach to detecting phishing web sites", *... conference on World Wide Web*, pp. 639–648, 2007, ISSN: 08963207. DOI: 10.1145/1242572.1242659. [Online]. Available: http://dl.acm.org/citation.cfm?id=1242659.

[14] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails", *Proceedings of the 16th international conference on World Wide Web - WWW '07*, p. 649, 2007, ISSN: 08963207. DOI: 10.1145/1242572.1242660. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1242572.1242660.

[15] R. Basnet, S. Mukkamala, and A. H. Sung, "Detection of phishing attacks: a machine learning approach", *Soft Computing Applications in Industry*, pp. 373–383, 2008. DOI: 10.1007/978-3-540-77465-5_19. [Online]. Available: http://link.springer.com/10.1007/978-3-540-77465-5%7B%5C_%7D19.

[16] "Linear regression", [Online]. Available: http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm.

[17] *Scikit-learn*. [Online]. Available: http://scikit-learn.org.

[18] *K-nearest neighbor*. [Online]. Available: http://www.scholarpedia.org/article/K-nearest%7B%5C_%7Dneighbor.

[19] A. Singh, N. Thakur, and A. Sharma, "A review of supervised machine learning algorithms", *Proceedings of the 10th INDIACom; 2016 3rd International Conference on Computing for Sustainable Global Development, INDIACom 2016*, pp. 1310–1315, 2016, ISSN: 0973-7529. [Online]. Available: https://www.engineeringvillage.com/share/document.url?mid=cpx%7B%5C_%7D4dbc01df158d4d5e77dM6b8b10178163171%7B%5C&%7Ddatabase=cpx.

[20] J. F. H., "Greedy function approximation: a gradient boosting machin", ISSN: 1098-6596. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[21] L. Ma, B. Ofoghi, P. Watters, and S. Brown, "Detecting phishing emails using hybrid features", *UIC-ATC 2009 - Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing in Conjunction with the UIC'09 and ATC'09 Conferences*, pp. 493–497, 2009. DOI: 10.1109/UIC-ATC.2009.103.

[22] A. A. Akinyelu and A. O. Adewumi, "Classification of phishing email using random forest machine learning technique", *Journal of Applied Mathematics*, vol. 2014, 2014, ISSN: 16870042. DOI: 10.1155/2014/425731.

[23] O. Akanbi, A. Abunadi, and A. Zainal, "Phishing website classification: a machine learning approach", *Journal of Information Assurance and Security*, vol. 9, no. September, pp. 354–366, 2014.

[24] H. B. Kazemian and S. Ahmed, "Comparisons of machine learning techniques for detecting malicious webpages", *Expert Systems with Applications*, vol. 42, no. 3, pp. 1166–1177, 2015, ISSN: 09574174. DOI: 10.1016/j.eswa.2014.08.046. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2014.08.046.

[25] H. Zhang, G. Liu, T. W. Chow, and W. Liu, "Textual and visual content-based anti-phishing: a bayesian approach", *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1532–1546, 2011, ISSN: 10459227. DOI: 10.1109/TNN.2011.2161999.

[26] H. Huang, L. Qian, and Y. Wang, "A svm-based technique to detect phishing urls", *Information Technology Journal*, vol. 11, no. 7, pp. 921–925, 2012, ISSN: 18125638. DOI: 10.3923/itj.2012.921.925. arXiv: 9809069v1 [arXiv:gr-qc]. [Online]. Available: http://docsdrive.com/pdfs/ansinet/itj/2012/921-925.pdf.

[27] S. L. Salzberg, "C4.5: programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993", *Machine Learning*, vol. 16, no. 3, pp. 235–240, Sep. 1994, ISSN: 1573-0565. DOI: 10.1007/BF00993309. [Online]. Available: https://doi.org/10.1007/BF00993309.

[28] Z. Gao, Y. Xu, F. Meng, F. Qi, and Z. Lin, "Improved information gain-based feature selection for text categorization", *2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace and Electronic Systems, VITAE 2014 - Co-located with Global Wireless Summit*, 2014. DOI: 10.1109/VITAE.2014.6934421.

[29] S. Afroz and R. Greenstadt, "Phishzoo: detecting phishing websites by looking at them", *Proceedings - 5th IEEE International Conference on Semantic Computing, ICSC 2011*, pp. 368–375, 2011. DOI: 10.1109/ICSC.2011.52.

[30] D. G. Lowe, "Object recognition from local scale-invariant features", in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[31] M. Hara, A. Yamada, and Y. Miyake, "Visual similarity-based phishing detection without victim site information", *2009 IEEE Symposium on Computational Intelligence in Cyber Security, CICS 2009 - Proceedings*, 2009. DOI: 10.1109/CICYBS.2009.4925087.

[32] H. M. A. Fahmy and S. A. Ghoneim, "Phishblock: a hybrid anti-phishing tool", *2011 International Conference on Communications, Computing and Control Applications, CCCA 2011*, pp. 2–6, 2011. DOI: 10.1109/CCCA.2011.6031523.

[33] A. K. Jain and B. B. Gupta, "Rule-based framework for detection of smishing messages in mobile environment", *Procedia Computer Science*, vol. 125, pp. 617–623, 2018, ISSN: 18770509. DOI: 10.1016/j.procs.2017.12.079.

[34] J. W. Joo, S. Y. Moon, S. Singh, and J. H. Park, "S-detector: an enhanced security model for detecting smishing attack for mobile computing", *Telecommunication Systems*, vol. 66, no. 1, pp. 29–38, 2017, ISSN: 15729451. DOI: 10.1007/s11235-016-0269-9.

[35] A. Rajaraman and J. D. Ullman, "Data mining", *Mining of Massive Datasets*, vol. 18 Suppl, pp. 114–142, 2011, ISSN: 00401706. DOI: 10.1007/978-1-4419-1280-0. arXiv: arXiv:1011.1669v3.

[36] G. James, D. Witen, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning with Applications in R*, 9-12. 2007, vol. 64, pp. 316–321, ISBN: 9780387781884. DOI: 10.1016/j.peva.2007.06.006. arXiv: arXiv:1011.1669v3. [Online]. Available: http://books.google.com/books?id=9tv0taI8l6YC.

[37] T. Fawcett, "An introduction to roc analysis", *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006, ISSN: 01678655. DOI: 10.1016/j.patrec.2005.10.010. arXiv: /dx.doi.org/10.1016/j.patrec.200 [http:].

[38] *Enron dataset site*. [Online]. Available: https://www.cs.cmu.edu/%7B~%7Denron/.

[39] B. Klimt and Y. Yang, "The enron corpus: a new dataset for email classification research", pp. 217–226, 2004, ISSN: 03029743. DOI: 10.1007/978-3-540-30115-8_22. [Online]. Available: http://link.springer.com/10.1007/978-3-540-30115-8%7B%5C_%7D22.

[40] D. Radev, *Clair collection of fraud email*, 2008. [Online]. Available: http://aclweb.org/aclwiki.

[41] *Fraudulent email corpus*. [Online]. Available: https://www.kaggle.com/rtatman/fraudulent-email-corpus.

[42] *Beautiful soup*. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/.

[43] "Email package", [Online]. Available: https://docs.python.org/2.7/library/email.html.

[44] "Dateparser", [Online]. Available: https://dateparser.readthedocs.io/en/latest/.

[45] "Datetime", [Online]. Available: https://docs.python.org/2.7/library/datetime.html%7B%5C#%7Dmodule-datetime.

[46] "Fuzzywuzzy", [Online]. Available: https://github.com/seatgeek/fuzzywuzzy.

[47] "Seaborn", [Online]. Available: https://seaborn.pydata.org/.

[48] A. Ronacher, *Flask*. [Online]. Available: http://flask.pocoo.org/.

[49] S. M. Ferrari, P. Fallahi, U. Politti, G. Materazzi, E. Baldini, S. Ulisse, P. Miccoli, A. Antonelli, D. Williams, E. Adn, J. E. Jackson, S. M. Sweeney, B. R. Haugen, E. K. Alexander, K. C. Bible, G. Doherty, S. J. Mandel, Y. E. Nikiforov, F. Pacini, G. Randolph, A. Sawka, M. Schlumberger, K. G. Schuff, S. I. Sherman, J. A. Sosa, D. Steward, R. M. Tuttle, and L. Wartofsky, *2015 American Thyroid Association Management Guidelines for Adult Patients with Thyroid Nodules and Differentiated Thyroid Cancer*, 1. 2015, vol. 26, pp. 4–25. DOI: 10.1158/1078-0432.CCR-12-2891.